

#2

**NONPROVISIONAL PATENT APPLICATION**

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

OLIFF & BERRIDGE, PLC  
P.O. Box 19928  
Alexandria, Virginia 22320  
Telephone: (703) 836-6400  
Facsimile: (703) 836-2787

Attorney Docket No.: 108401

Date: January 22, 2001

**BOX PATENT APPLICATION**

Customer Number: 25944

**NONPROVISIONAL APPLICATION TRANSMITTAL  
RULE §1.53(b)**

Director of the U.S. Patent and Trademark Office  
Washington, D.C. 20231

Sir:

Transmitted herewith for filing under 37 C.F.R. §1.53(b) is the nonprovisional patent application

For (Title): STRUCTURED DOCUMENT PROCESSING SYSTEM AND STRUCTURED  
DOCUMENT PROCESSING METHOD

By (Inventors): Setsu KUNITAKE, Ichiro YAMASHITA and Shigehisa KAWABE



- ☒ Formal drawings (Figs. 1-27; 33 sheets) are attached.  
☒ A Declaration and Power of Attorney is filed herewith.  
☒ An assignment of the invention to FUJI XEROX CO., LTD. is filed herewith.  
☐ An Information Disclosure Statement is filed herewith.  
☐ Entitlement to small entity status is hereby asserted.  
☐ A Preliminary Amendment is filed herewith.  
☐ Please amend the specification by inserting before the first line the sentence --This nonprovisional application claims the benefit of U.S. Provisional Application No. \_\_\_\_\_, filed \_\_\_\_\_.--  
☒ Priority of foreign application No. 2000-018059 filed January 25, 2000 in JAPAN is claimed (35 U.S.C. §119).  
☒ A certified copy of the above corresponding foreign application is filed herewith.  
☒ The filing fee is calculated below:

**CLAIMS IN THE APPLICATION AFTER ENTRY OF  
ANY PRELIMINARY AMENDMENT NOTED ABOVE**

FOR:	NO. FILED	NO. EXTRA
BASIC FEE		
TOTAL CLAIMS	14 - 20	= 0
INDEP CLAIMS	5 - 3	= 2
<input type="checkbox"/> MULTIPLE DEPENDENT CLAIMS PRESENTED		

\* If the difference is less than zero, enter "0".

**SMALL ENTITY**

RATE	FEE
	\$ 355
x 9 =	\$
x 40 =	\$
+ 135 =	\$
TOTAL	\$

OR

OR

OR

OR

OR

OR

**OTHER THAN A  
SMALL ENTITY**

RATE	FEE
	\$ 710
x 18	\$
x 80	\$ 160
+ 270	\$
TOTAL	\$ 870

- ☒ Check No. 115621 in the amount of \$870.00 to cover the filing fee is attached. Except as otherwise noted herein, the Director is hereby authorized to charge any other fees that may be required to complete this filing, or to credit any overpayment, to Deposit Account No. 15-0461. Two duplicate copies of this sheet are attached.

Respectfully submitted,

James A. Oliff  
Registration No. 27,075

Thomas J. Pardini  
Registration No. 30,411

JAO:TJP/zmc

日 本 国 特 許 庁

PATENT OFFICE  
JAPANESE GOVERNMENT



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

2000年 1月25日

出 願 番 号

Application Number:

特願2000-018059

出 願 人

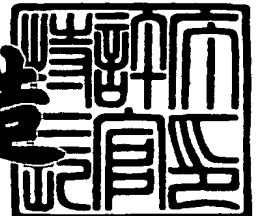
Applicant (s):

富士ゼロックス株式会社

2000年10月27日

特許庁長官  
Commissioner,  
Patent Office

及川耕造



出証番号 出証特2000-3089161

【書類名】 特許願  
【整理番号】 FN99-00532  
【提出日】 平成12年 1月25日  
【あて先】 特許庁長官殿  
【国際特許分類】 G06F 17/27  
【請求項の数】 14  
【発明者】

【住所又は居所】 神奈川県足柄上郡中井町境 4 3 0 グリーンテクなかい  
富士ゼロックス株式会社内

【氏名】 國武 節

【発明者】

【住所又は居所】 神奈川県足柄上郡中井町境 4 3 0 グリーンテクなかい  
富士ゼロックス株式会社内

【氏名】 山下 一郎

【発明者】

【住所又は居所】 神奈川県足柄上郡中井町境 4 3 0 グリーンテクなかい  
富士ゼロックス株式会社内

【氏名】 川邊 恵久

【特許出願人】

【識別番号】 000005496

【氏名又は名称】 富士ゼロックス株式会社

【電話番号】 0462-38-8516

【代理人】

【識別番号】 100086531

【弁理士】

【氏名又は名称】 澤田 俊夫

【電話番号】 03-5541-7577

【手数料の表示】

【予納台帳番号】 038818

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 構造化文書処理システム及び構造化文書処理方法

【特許請求の範囲】

【請求項 1】

構造的に表現され 1 以上の文書部品を含んだ構造化文書进行处理するための構造化文書処理システムであって、

文書部品の抽出を指定する抽出命令又は文書部品の所定回数の複写を指定する反復複写命令又は文書部品の挿入若しくは置換を指定する挿入置換命令を該当する文書部品に付与する命令付与手段と、

構造化文書の構造を解析して構文解析木を生成する解析手段と、

前記解析手段により生成された構文解析木を基に構造化文書に付与された命令と文書部品を分離して命令を取り出す命令分離手段と、

抽出命令が付与された第 1 の構造化文書から前記命令分離手段によって取り出された抽出命令と、反復複写命令及び／又は挿入置換命令が付与された第 2 の構造化文書から前記命令分離手段によって取り出された反復複写命令及び／又は挿入置換命令とを併合・整列して、該第 1 及び第 2 の構造化文書进行处理するための命令列からなる文書処理記述を生成する文書処理記述生成手段と、

抽出命令の対象である第 1 の文書部品を第 1 の構造化文書から取り出す抽出手段と、

反復複写命令の対象である文書部品と該文書部品付与された命令を所定回数だけ反復複写する反復複写手段と、

第 1 の文書部品を、挿入置換命令の対象である第 2 の文書部品の前又は後に挿入するか又は該第 2 の文書部品と置換する挿入置換手段と、

文書処理記述を順次解釈して、前記抽出手段、前記反復複写手段、及び／又は前記挿入置換手段を用いて文書部品を合成するインタプリタと、  
を具備することを特徴とする構造化文書処理システム。

【請求項 2】

前記抽出命令は文書部品の属性の取り出しを指定する属性抽出命令であり、

前記挿入置換命令は文書部品の属性の置換を指定する属性置換命令であり、

前記命令分離手段は、構造化文書から属性抽出命令と属性置換命令を取り出し

前記挿入置換手段は、第 1 の文書部品の属性を属性置換命令の対象である第 2 の文書部品の属性と置換する属性置換手段であり、

前記インタプリタは、前記文書処理記述合成手段が合成した文書処理記述を解釈して、属性抽出命令で指定された文書部品の所定のノードの属性を抽出するとともに、該抽出された属性を属性置換命令で指定された文書部品の所定のノードに設定する、

ことを特徴とする請求項 1 に記載の構造化文書処理システム。

#### 【請求項 3】

前記属性置換手段は、属性置換命令に対して、文書部品にあらかじめ設定されている属性値文字列とシステムの状態から得られる文字列を組み合わせて合成される文字列に置き換える属性置換手段であることを特徴とする請求項 2 に記載の構造化文書処理システム。

#### 【請求項 4】

前記抽出命令はパス名付き抽出命令であり、

前記反復複写命令及び挿入置換命令はそれぞれパターン式付き反復複写命令及びパターン式付き挿入置換命令であり、

前記命令分離手段は、構造化文書から、抽出命令とパス名を取り出すか、又は、反復複写命令と挿入置換命令とパターン式を取り出し、

前記反復複写手段は、抽出した文書部品に付与されたパス名とパターン式とのパターン・マッチングを行い、マッチするパス名を持つ文書部品の個数だけ反復複写を実行し、

前記挿入置換手段は、抽出した文書部品に付与されたパス名とパターン式とのパターン・マッチングを行い、マッチするパス名を持つ文書部品を挿入又は置換する、

ことを特徴とする請求項 1 に記載の構造化文書処理システム。

#### 【請求項 5】

前記挿入置換手段は、前記抽出手段によって取り出された文書部品に付与され

た抽出命令を挿入又は置換することを特徴とする請求項 1 に記載の構造化文書処理システム。

【請求項 6】

前記挿入置換手段は、前記抽出手段によって取り出された文書部品に付与された抽出命令を挿入又は置換する際に、パス名付き抽出命令のパス名を変更してから挿入置換することを特徴とする請求項 6 に記載の構造化文書処理システム。

【請求項 7】

ネットワーク接続された 2 以上のコンピュータで構成される分散ネットワークシステム上におけるコンピュータ通信による協調的な処理によって実現される、構造的に表現され 1 以上の文書部品を含んだ構造化文書処理システムであって、

構造化文書を所定フォーマットのファイルとして格納するとともに、ファイル名を受信したことに応答して該当するファイルを前記ネットワーク経由で送信するファイル・サーバと、ファイルに対して文書処理を行う構造化文書処理サーバとを少なくとも含み、前記構造化文書処理サーバは、

文書部品の抽出を指定した抽出命令が付与された第 1 の構造化文書のファイル名と反復複写命令又は挿入置換命令を付与した第 2 の構造化文書のファイル名を含む処理起動記述を入力し解析して、該処理起動記述に含まれるファイル名を前記ファイル・サーバに前記ネットワーク経由で送信し、ファイル名に該当する各ファイルを前記ファイル・サーバからネットワーク経由で入力する入力手段と、

第 1 の構造化文書及び第 2 の構造化文書を解析して構文解析木を生成し、構文解析木を探索して文書部品と命令とを分離して命令を取り出し、各命令を併合・整列して構造化文書进行处理するための命令列からなる文書処理記述を生成し、該文書処理記述を解釈して構造化文書を合成処理する文書処理手段と、

前記文書処理手段で処理して得られた構造化文書又は文書部品を、所定フォーマットのファイルとしてネットワーク経由で出力する出力手段とを具備することを特徴とする構造化文書処理システム。

【請求項 8】

処理起動記述は構造化文書処理サーバのサーバ名を含む形式で前記ネットワー

ク上の分散ファイル名を規定することができ、

前記ネットワーク上には少なくとも構造化文書処理する第1及び第2の構造化文書処理サーバが存在し、

第1の構造化文書処理サーバに入力される第1の処理起動記述では、文書処理の対象となる第1の原料文書及び／又は第1の雛型文書のファイル名が第2の構造化文書処理サーバのサーバ名を含む第2の処理起動記述の形式で記述され、

第1の構造化文書処理サーバは、第1の処理起動記述を入力したことに応答して、第1の原料文書及び／又は第1の雛型文書のファイル名として記述された第2の処理起動記述を抜き出して第2の構造化文書処理サーバに前記ネットワーク経由で送信するとともに、第2の構造化文書処理サーバが第2の処理起動記述を起動して出力する構造化文書又は文書部品を含むファイルをネットワーク経由で受信して第1の原料文書及び／又は第1の雛型文書として使用する、ことを特徴とする請求項7に記載の構造化文書処理システム。

【請求項9】

第2の処理起動記述を入力する第2の構造化文書処理サーバは、前記ネットワーク経由の通信を必要としない第1の構造化文書処理サーバと同一のコンピュータ・システム上で構成され、

第1の構造化文書処理サーバは、構造化文書又は文書部品を含むファイルに替えて、第2の構造化文書処理サーバによる処理結果である構造化文書又は文書部品を構文解析木として入力する切り替え手段を具備する、ことを特徴とする請求項8に記載の構造化文書処理システム。

【請求項10】

前記ファイル・サーバから入力した原料文書又は雛型文書の構文解析木をファイル名又は処理起動記述と対応付けて保持する保持手段と、

ファイル名に対応する構造化文書ファイルを前記ファイル・サーバから入力する代わりに、該当する構文解析木を前記保持手段から入力する入力手段と、をさらに具備することを特徴とする請求項7に記載の構造化文書処理システム。

【請求項11】

前記構造化文書処理システムは、文書部品の抽出を指定した抽出命令が付与さ



れた第 1 の構造化文書のファイル名と反復複写命令又は属性置換命令を付与した第 2 の構造化文書のファイル名を含む処理起動記述を入力し、

前記属性置換手段は、前記処理起動記述の一部を、文書部品にあらかじめ設定されている属性文字列と、置き換えて得られる文字列を、該文書部品の属性文字列として設定することを特徴とする請求項 2 に記載の構造化文書処理システム。

【請求項 1 2】

構造的に表現され 1 以上の文書部品を含んだ構造化文書进行处理するための構造化文書処理システムであって、

構造化文書の構造を解析して構文解析木を生成する解析手段と、

前記解析手段により生成された構文解析木を基に構造化文書に付与された命令と文書部品を分離して命令を取り出すとともに、命令の文法的なエラーを検出してエラー情報を出力する命令分離手段と、

エラー情報を入力してエラー通知を行なう文書を合成するエラー通知文書合成手段と、

構造化文書から取り出された各命令を併合・整列して構造化文書进行处理するための命令列からなる文書処理記述を生成するとともに、エラー通知文書へのアクセス情報を生成する処理起動記述合成手段と、

処理起動記述を解釈して、エラー通知文書を取り出す処理起動記述解析手段と

エラー通知文書を保持する保持手段と、

を具備することを特徴とする構造化文書処理システム。

【請求項 1 3】

構造的に表現され 1 以上の文書部品を含んだ構造化文書进行处理するための構造化文書処理方法であって、

命令が付与された構造化文書を解析して構文解析木を生成するステップと、

構文解析木を探索して、文書部品と命令とを分離して命令を取り出すステップと、

構造化文書から取り出された各命令を併合・整列して、構造化文書进行处理するための命令列からなる文書処理記述を生成するステップ、

文書処理記述を解釈して、構造化文書を合成処理するステップと、  
を具備することを特徴とする構造化文書処理方法。

【請求項 1 4】

構造的に表現され 1 以上の文書部品を含んだ構造化文書进行处理するための構造化文書処理方法であって、

(a) 文書部品の抽出を指定する抽出命令が付与された第 1 の構造化文書を解析して構文解析木を生成するステップと、

(b) 文書部品の所定回数の複写を指定する反復複写命令又は文書部品の挿入若しくは置換を指定する挿入置換命令が付与された第 2 の構造化文書を解析して構文解析木を生成するステップと、

(c) 構文解析木を探索して、文書部品と命令とを分離して命令を取り出すステップと、

(d) 抽出命令が付与された第 1 の構造化文書から取り出された抽出命令と、反復複写命令及び／又は挿入置換命令が付与された第 2 の構造化文書から取り出された反復複写命令及び／又は挿入置換命令とを併合・整列して、該第 1 及び第 2 の構造化文書进行处理するための命令列からなる文書処理記述を生成するステップ

(e) 文書処理記述を解釈して、抽出命令の対象である第 1 の文書部品を第 1 の構造化文書から取り出すステップと、

(f) 文書処理記述を解釈して、反復複写命令の対象である文書部品と該文書部品付与された命令を所定回数だけ反復複写するステップと、

(g) 文書処理記述を解釈して、第 1 の文書部品を、挿入置換命令の対象である第 2 の文書部品の前又は後に挿入するか又は該第 2 の文書部品と置換するステップと、

(h) 前記ステップ (e) 乃至 (g) の結果得られた構文解析木を出力するステップと、

を具備することを特徴とする構造化文書処理方法。

【発明の詳細な説明】

【0 0 0 1】

## 【発明の属する技術分野】

本発明は、文書を章、節、段落、図表、あるいは表題や章題、概要などの複数の素片に分解し各素片をノードとするツリー構造やグラフ構造などのように構造的に表現して取り扱う構造化文書の処理技術に係り、特に、複数の構造化文書を基にして新たに文書を合成する構造化文書の処理技術に関する。

## 【0002】

更に詳しくは、本発明は、複数の構造化文書から特定の条件を満たす文書部分（「文書部品」）を取り出すとともに文書部品を別の文書中に挿入又は置換して合成を行なう構造化文書の処理技術に係り、構造化文書から文書部品を抽出したり、各文書部品を雛型文書中に挿入又は置換したりする手続きを記述したスクリプトを用いることなしに文書の合成処理を行う構造化文書の処理技術に関する。

## 【0003】

## 【従来の技術】

一般に、文書がテキスト文字列のみで構成されることは稀であり、章や節、段落という区切りを持ったり、図表などのコンテンツが挿入されていたり、あるいは表題や章題、概要などの素片を含むことが多い。

## 【0004】

このため、単一の文書を章、節、段落、図表、あるいは表題や章題、概要などの複数の素片に分解し、各素片をノードとするツリー構造やグラフ構造などのように構造的に表現して取り扱う文書処理技術が開発されてきた。このように構造化された文書は一般に「構造化文書」と呼ばれ、計算機システムを用いてさまざまな処理を行うことが可能である。

## 【0005】

構造化文書では、各ノードとリンクで表現される親子関係は、文書の論理的な構造を表現する。例えば「章題」「図」「章」などの属性が付与されたノードに対して、印刷のため各ノードの単位でレイアウト処理を行なったり、版下作成の処理や、ノードの属性を手がかりにして抄録集や目次を自動的に作成する処理などが可能になる。

## 【0006】

現在、構造化文書を記述するフォーマットとしては、SGML (Standard Generalized Markup Language) やXML (eXtensible Markup Language) HTML (Hyper Text Markup Language) などの記述言語がよく知られている。例えば、HTMLは、表 (TABLE) や箇条書き (UL) を指示する記法がある。

## 【0007】

プレーンなテキスト・ファイルに構造を指示する方法の一つとして、マークアップと呼ばれる方法がある。マークアップは、あらかじめ決められた規則に基づき、特定の論理的な構造の開始を表すタグ記号と終了を表すタグ記号で文書の一部を挟むことで構造を定める。例えばHTMLでは、箇条書きの開始タグを<UL>とし、終了タグを</UL>として、箇条を立てる項目の開始タグを<LI>終了タグを</LI>とすると、次のように表記される。

## 【0008】

## 【数1】

```
<UL>
<LI>箇条書き項目1</LI>
<LI>箇条書き項目2</LI>
<LI>箇条書き項目N</LI>
</UL>
```

## 【0009】

このような構造化文書に対して、文書の構造を利用した検索を行なうと、複数の文書から特定の条件を満たす文書部分（以下では、「文書部品」と呼ぶ）を取り出し、取り出した文書部品を別の文書中に挿入して合成を行なう方式が提案されている。例えば、本出願人に既に譲渡されている特開平6-52161号公報に「文書処理方法及び文書処理装置」として開示される文書処理方式では、ツリーまたはグラフで表現された構造化文書に対して、ノードとして表される文書部品（同公報中では「構成子」と呼ばれている）のタイプを指定して検索する select-type 命令によって、所定の属性を持つ文書部品を取り出して、第

2の文書に挿入することが記述されている。例えば `select-type` 命令の引数として、' `Figure` ' や ' `Segment` ' を指定することで、それぞれ図と節を検索して取り出す例が開示されている。また、文書部品の挿入に関しては、挿入先の文書の特定のノードを指定し、指定したノードの末子に挿入する例が記述されている。また、ノードの親子関係を判定して、これによって特定の構造パターンを有するノードを検索することができる。

## 【0010】

富士ゼロックス株式会社の製品 " `Akane` " は、ウィンドウ・システム上で動作する構造化文書エディタを中心とした文書処理アプリケーション・ソフトウェアである。 `Akane` の周辺ツールとして、文書処理コマンド・セットが提供されている。「 `Akane` 文書操作コマンドセットプログラマーズガイド」第3章応用事例 (p. 2-95~2-96) には、文書の論理構造をパターンで指定して、特定の条件を満たす構造を有するノードを文書部品として取り出し、一つの文書として合成する例が記述されている。

## 【0011】

このように、あらかじめ検索式を指定したプログラムと、検索結果として取り出された文書部品を処理するプログラムをパイプラインで結合し、入力とする原料文書から、特定の条件を満たす文書部品を取り出して新たな構造化文書や文書部品を合成することができる。

## 【0012】

XMLが構造化文書を記述可能な言語であることは既に述べた通りである。「XML開発事例」(株式会社ASCII ISBN4-7561-3112-3) には、XMLで記述された構造化文書を入力し、処理を行なうXSL (`eXtensible Stylesheet Language`) 言語とそのプロセッサによる構造化文書の扱いが開示されている。XSLの構文は、例えば次のような構造をとる。

## 【0013】

【数2】

<rule>

```

    [ pattern ]
    [ action ]
</rule>

```

【0014】

ここで、[ pattern ] は、処理の対象とする文書部品の検索式を記述する。また、[ action ] は、検索された文書部品に対する処理を記述する。検索式の例を以下に示す。

【0015】

【数3】

```

<rule>
    <target-element type = " section" />
    <element type = " figure" />
    [ action ]
</rule>

```

【0016】

<target-element type=" section" />は、検索する文書部品のノードの型が" section" であり、次の<element type = " figure" >は、文書部品のノードの型が" figure" である子ノードを含むように限定することを示す検索式である。

【0017】

また、次に示す例は、ノードの型が" employee" であって、かつ、親ノードの型が" person" であるような文書部品を検索する式である。

【0018】

【数4】

```

<rule>
    <element type = " employee" />
    <target-element type = " person" />
    [ action ]
</rule>

```

## 【 0 0 1 9 】

このように、検索式と、検索結果として取り出された文書部品を処理するアクションとをあらかじめ記述したスクリプトを解釈実行することで、入力とする原料文書から、特定の条件を満たす文書部品を取り出して新たな構造化文書や文書部品を合成することができる。

## 【 0 0 2 0 】

また、本出願人に既に譲渡されている特開平 7 - 5 6 9 2 0 号公報において「構造化文書処理装置」として開示される文書処理装置は、構造化文書から複数の文書部品を抽出して列にして出力する部分構造列抽出部と、文書部品の列を入力して処理を行う処理実行部を備えている。かかる構成の文書処理装置によれば、抽出指示部と処理指示部を別々に管理することで、原料とする文書の構造の変更に対して、文書処理の変更を抽出指示部だけにとどめることができる。例えば、上記した各従来技術に対してシステムや文書の改変と維持管理が簡単になる。

## 【 0 0 2 1 】

## 【発明が解決しようとする課題】

上述した従来技術は、要するに図 2 6 又は図 2 7 のようなシステム構成となる。例えば、図 2 6 に示す構造文書処理システムの場合、まず、原料文書と雛型文書（テンプレート）が抽出／合成プログラムに投入される。抽出／合成プログラムは、原料文書からの文書部品の抽出や雛型文書への文書部品の挿入・置換などの手続きをスクリプト形式で記述した「抽出／合成スクリプト」に従って処理を実行し、合成文書を生成する。

## 【 0 0 2 2 】

また、図 2 7 に示す構造文書処理システムの場合、まず、抽出プログラムが抽出スクリプトで記述された手続きに従い原料プログラムから文書部品の抽出処理を行う。抽出された各文書部品は、雛型文書とともに合成プログラムに投入される。そして、合成プログラムは、合成スクリプトで記述された手続きに従って各文書部品の雛型文書への挿入・置換処理を実行して、合成文書を生成する。

## 【 0 0 2 3 】

これら上記した技術はいずれも、原料文書から文書部品を抽出するための手続

きを記述したスクリプト（抽出スクリプト）や、結果として出力する文書の基になる雛形文書に対して、原料文書から抽出した文書部品を挿入または置換するためのスクリプト（合成スクリプト）を使用するものである。言い換えれば、これら従来技術は、元の文書とは別にこれらスクリプトを管理する必要がある、以下に示す問題を包含する。

## 【0024】

1. 原料から文書部品を取り出すため、文書部品の構造やパターンを特定する検索式をスクリプト中に記述する必要がある。したがって、原料文書の構造を変更すると、スクリプト中の検索式もこれに合わせて変更しなくてはならない。

## 【0025】

2. 構造が異なる複数の原料文書を混在させて処理するには、異なる構造ごとにスクリプトを用意しなくてはならない。

## 【0026】

3. 文書部品の構造やパターンを特定する検索式や、取り出した文書部品を処理する手順を手続きとして記述することが難しい。一般には、「条件Aを満たす文書部品を検索して、得られた結果に処理Bを行なう」という手続き的な指示をスクリプトとして記述する必要がある。また、文書部品の数が原料文書によって変わる場合に、文書部品の数を数え、繰り返しを行なう `for` 文や `repeat` 文などの繰り返し命令と、所望の処理を実際に行なう命令（例えば挿入命令）とを組み合わせ、繰り返し処理（例えば挿入や置換）を行なう指示をスクリプトとして記述する必要がある。このような記述やスクリプトの作成には、プログラミングの相当程度の知識が必要であり、一般ユーザが広く行なうことは難しく、あまり普及しないと思料される。

## 【0027】

4. 文書処理の中間結果（例えば検索処理によって抽出された文書部品）を簡単に再利用する機構について言及していない。例えば、上述した `Akane` の場合には、中間結果をファイルに保存するスクリプトを明示的に記述する必要がある。

## 【0028】



第 1 の問題点に関しては、原料文書の構造を変更した際に、これを処理するスクリプトを網羅的に探したり、スクリプトの変更を忘れると文書処理が動作不良を起こすなどの不具合が発生する。

【 0 0 2 9 】

また、第 2 の問題点に関しては、構造が相違する入力文書ごとにそれを処理する専用のスクリプトの開発とその維持管理に手間や労力が掛かるという不具合がある。また、利用者にとっては、適切なスクリプトを選んで利用する必要がある。スクリプトの選択を誤ると、文書処理が正しく動作しないなどの不具合が発生する。

【 0 0 3 0 】

また、第 3 の問題点に関しては、それぞれの目的に適合した構造化文書処理のアプリケーションを作成することができる文書処理システムを、利用者自らが実装することが極めて困難となる。

【 0 0 3 1 】

また、第 4 の問題点に関しては、効率の良いアプリケーションの開発に手間がかかるという不具合がある。

【 0 0 3 2 】

本発明は上述のような技術的課題を勘案したものであり、その目的は、文書を章、節、段落、図表、あるいは表題や章題、概要などの複数の素片に分解し各素片をノードとするツリー構造やグラフ構造などのように構造的に表現して取り扱うことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することにある。

【 0 0 3 3 】

本発明の更なる目的は、複数の構造化文書を基にして新たに文書を合成することができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することにある。

【 0 0 3 4 】

本発明の更なる目的は、複数の構造化文書から特定の条件を満たす文書部分（「文書部品」）を取り出すとともに文書部品を別の文書中に挿入又は置換して合

成を行なうことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することにある。

【 0 0 3 5 】

本発明の更なる目的は、構造化文書から文書部品を抽出したり、各文書部品を雛型文書中に挿入又は置換したりする手続きを記述したスクリプトを用いることなしに構造化文書の合成処理を行うことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することにある。

【 0 0 3 6 】

【課題を解決するための手段】

本発明は、上記課題を参酌しなされたものであり、その第 1 の側面は、構造的に表現され 1 以上の文書部品を含んだ構造化文書进行处理するための構造化文書処理システムであって、

文書部品の抽出を指定する抽出命令又は文書部品の所定回数の複写を指定する反復複写命令又は文書部品の挿入若しくは置換を指定する挿入置換命令を該当する文書部品に付与する命令付与手段と、

構造化文書の構造を解析して構文解析木を生成する解析手段と、

前記解析手段により生成された構文解析木を基に構造化文書に付与された命令と文書部品を分離して命令を取り出す命令分離手段と、

抽出命令が付与された第 1 の構造化文書から前記命令分離手段によって取り出された抽出命令と、反復複写命令及び／又は挿入置換命令が付与された第 2 の構造化文書から前記命令分離手段によって取り出された反復複写命令及び／又は挿入置換命令とを併合・整列して、該第 1 及び第 2 の構造化文書进行处理のための命令列からなる文書処理記述を生成する文書処理記述生成手段と、

抽出命令の対象である第 1 の文書部品を第 1 の構造化文書から取り出す抽出手段と、

反復複写命令の対象である文書部品と該文書部品付与された命令を所定回数だけ反復複写する反復複写手段と、

第 1 の文書部品を、挿入置換命令の対象である第 2 の文書部品の前又は後に挿入するか又は該第 2 の文書部品と置換する挿入置換手段と、

文書処理記述を順次解釈して、前記抽出手段、前記反復複写手段、及び／又は前記挿入置換手段を用いて文書部品を合成するインタプリタと、を具備することを特徴とする構造化文書処理システムである。

## 【 0 0 3 7 】

本発明の第 1 の側面に係る構造化文書処理システムは、命令付与手段によって文書部品の抽出を指定する抽出命令や、文書部品の所定回数の複写を指定する反復複写命令、文書部品の挿入若しくは置換を指定する挿入置換命令を、構造化文書中の該当する文書部品に付与することができる。

## 【 0 0 3 8 】

また、この構造化文書処理システムに対して、抽出命令が付与された第 1 の構造化文書と、反復複写命令及び／又は挿入置換命令が付与された第 2 の構造化文書の各々を投入すると、解析手段によって各構造化文書毎に文書解析木が構成され、命令分離手段によって各構造化文書に付与された命令列を取り出される。さらに、取り出された複数の命令列の併合・整列、及び変換が行なわれ、文書処理記述が文書処理記述合成手段によって合成される。

## 【 0 0 3 9 】

インタプリタは、文書処理記述を順に走査して、文書処理記述中に含まれる各命令に従って抽出手段によって第 1 の文書部品を第 1 の文書から抽出し、反復複写手段によって第 2 の構造化文書の文書部品を第 1 の文書部品の数に依存して定まる回数だけ反復複写し、また、挿入置換手段によって第 2 の構造化文書に第 1 の文書部品を挿入又は置換することで、構造化文書の合成を行なうことができる。

## 【 0 0 4 0 】

本発明の第 1 の側面に係る構造化文書処理システムによれば、以下に示す作用効果を奏することができる。すなわち、

## 【 0 0 4 1 】

1. 原料文書の構造を変更した際に、これを処理するための別に管理されるスクリプトを用意しなくてよい。
2. 構造が異なる複数の原料文書を混在させて文書部品を取り出すには、各文

書毎に文書部品に抽出命令を付与することで行なうので、文書毎の抽出の処理を個別に指定しなくてよい。同様に、文書部品を挿入又は置換する処理の指定も不要である。また、ユーザは、文書の構造が変わる度に適切なスクリプトを指定する必要がない。

3. 文書部品は、所望の命令を元の文書中に直接的に付与することと、取り出した文書部品と挿入置換される文書部品の条件の指定（例えばラベル文字列が一致するなど）によって作成することができる。また、各ユーザがそれぞれの目的に応じて雛型文書をデザイン（作成）したり、所望の処理を行なう命令が組み込まれた雛型文書を選択して、ユーザ自らが原料文書と組み合わせることで文書処理アプリケーションを構成することができる。このように、手続き的でない、言い換えれば宣言的な方法で文書処理を構成することができるので、プログラミングの知識を十分に持たないユーザが広く文書処理を行なうことができる。すなわち、各ユーザがそれぞれの目的に合わせた構造化文書処理のアプリケーションを容易に作成することができる構造化文書処理システムが実現できる。

#### 【0042】

本発明の第1の側面に係る構造化文書処理システムにおいて、前記抽出命令は文書部品の属性の取り出しを指定する属性抽出命令であり、また、前記挿入置換命令は文書部品の属性の置換を指定する属性置換命令であってもよい。このような場合、前記命令分離手段は構造化文書から属性抽出命令と属性挿入置換命令を取り出し、前記挿入置換手段は第1の文書部品の属性を属性置換命令の対象である第2の文書部品の属性と置換する属性置換手段であるとともに、前記インタプリタは前記文書処理記述合成手段が合成した文書処理記述を解釈して属性抽出命令で指定された文書部品の所定のノードの属性を抽出するようにしてもよい。この結果、該抽出された属性を属性置換命令で指定された文書部品の所定のノードに設定することができる。

#### 【0043】

この場合、前記属性置換手段は、属性置換命令に対して、文書部品にあらかじめ設定されている属性値文字列とシステムの状態から得られる文字列を組み合わせる文字列に置き換える属性置換手段であってもよい。

## 【 0 0 4 4 】

あるいは、前記抽出命令はパス名付き抽出命令であり、また、前記反復複写命令及び挿入置換命令はそれぞれパターン式付き反復複写命令及びパターン式付き挿入置換命令であってもよい。このような場合、前記命令分離手段は、構造化文書から、抽出命令とパス名を取り出すか、又は、反復複写命令と挿入置換命令とパターン式を取り出し、前記反復複写手段は、抽出した文書部品に付与されたパス名とパターン式とのパターン・マッチングを行い、マッチするパス名を持つ文書部品の個数だけ反復複写を実行し、また、前記挿入置換手段は、抽出した文書部品に付与されたパス名とパターン式とのパターン・マッチングを行い、マッチするパス名を持つ文書部品を挿入又は置換するようにすればよい。

## 【 0 0 4 5 】

また、前記挿入置換手段は、前記抽出手段によって取り出された文書部品に付与された抽出命令を挿入又は置換するようにしてもよい。この結果、構造化文書処理システムを複数段接続してパイプラインを構成することができる。この機能によって、より柔軟で複雑な構造化文書処理を実現することが可能となる。

## 【 0 0 4 6 】

このような場合、前記挿入置換手段は、抽出命令を挿入又は置換する際に、パス名付き抽出命令のパス名を変更してから挿入置換するようにしてもよい。この結果、最初の構造化文書処理システムで出力された原料文書に含まれている文書部品は、2 段目以降の構造化文書処理ではパス名を調べることで識別することができる。

## 【 0 0 4 7 】

また、本発明の第 2 の側面は、ネットワーク接続された 2 以上のコンピュータで構成される分散ネットワーク・システム上におけるコンピュータ通信による協調的な処理によって実現される、構造的に表現され 1 以上の文書部品を含んだ構造化文書処理システムであって、

構造化文書を所定フォーマットのファイルとして格納するとともに、ファイル名を受信したことに応答して該当するファイルを前記ネットワーク経由で送信するファイル・サーバと、ファイルに対して文書処理を行う構造化文書処理サーバ

とを少なくとも含み、前記構造化文書処理サーバは、

文書部品の抽出を指定した抽出命令が付与された第1の構造化文書のファイル名と反復複写命令又は挿入置換命令を付与した第2の構造化文書のファイル名を含む処理起動記述を入力し解析して、該処理起動記述に含まれるファイル名を前記ファイル・サーバに前記ネットワーク経由で送信し、ファイル名に該当する各ファイルを前記ファイル・サーバからネットワーク経由で入力する入力手段と、

第1の構造化文書及び第2の構造化文書を解析して構文解析木を生成し、構文解析木を探索して文書部品と命令とを分離して命令を取り出し、各命令を併合・整列して構造化文書进行处理するための命令列からなる文書処理記述を生成し、該文書処理記述を解釈して構造化文書を合成処理する文書処理手段と、

前記文書処理手段で処理して得られた構造化文書又は文書部品を、所定フォーマットのファイルとしてネットワーク経由で出力する出力手段とを具備することを特徴とする構造化文書処理システムである。

#### 【0048】

ここで言うネットワークとは、例えば、インターネットのように複数のコンピュータがTCP/IP (Transmission Control Protocol / Internet Protocol) 接続して構成される分散ネットワークである。このような場合、構造化文書は、例えばHTML (Hyper Text Markup Language) のようなマークアップ言語によって記述される。また、ファイル・サーバは、HTTP (Hyper Text Transfer Protocol) プロトコルに従ってHTMLファイルを提供するHTTPサーバとして構成される。また、ファイル名は、URL (Uniform Resource Locator) のような資源アクセス識別子の形式で表記される。

#### 【0049】

また、構造化文書処理サーバは、HTTPリクエストの形式で処理起動記述を入力することができる。この処理起動記述は、HTTPリクエストのURL中に別のURLを埋め込んだ記述形式の「ヴァーチャルURL」として構成される。より具体的には、第1の構造化文書及び第2の構造化文書それぞれのファイル名

、すなわちURLを含んだヴァーチャルURLである。したがって、構造化文書処理サーバは、このようなヴァーチャルURLを基にして、該当するファイル・サーバすなわちHTTPサーバから第1及び第2の構造化文書をネットワーク経由で取得することができる。

## 【0050】

さらに、構造化文書処理サーバは、第1の構造化文書及び第2の構造化文書を解析してそれぞれの構文解析木を生成し、構文解析木を探索して文書部品と命令とを分離して命令を取り出し、各命令を併合・整列して構造化文書进行处理するための命令列からなる文書処理記述を生成し、該文書処理記述を解釈して、構造化文書を合成処理することができる。

## 【0051】

このようにして得られた構造化文書又は文書部品は、HTMLフォーマットのファイルとして、すなわちHTTPレスポンスとしてネットワーク経由で出力することができる。

## 【0052】

本発明の第2の側面に係る構造化文書処理システムでは、処理起動記述は構造化文書処理サーバのサーバ名を含む形式で前記ネットワーク上の分散ファイル名を規定することを許容してもよい。

## 【0053】

例えば、前記ネットワーク上に構造化文書进行处理する第1及び第2の構造化文書処理サーバが存在する場合、第1の構造化文書処理サーバに入力される第1の処理起動記述において、文書処理の対象となる第1の原料文書及び／又は第1の雛型文書のファイル名を第2の構造化文書処理サーバのサーバ名を含む第2の処理起動記述の形式で記述することができる。

## 【0054】

第1の構造化文書処理サーバは、第1の処理起動記述を入力したことに応答して、第1の原料文書及び／又は第1の雛型文書のファイル名として記述された第2の処理起動記述を抜き出して、これを第2の構造化文書処理サーバに前記ネットワーク経由で送信すればよい。第2の構造化文書処理サーバは、第2の処理起

動記述を解釈して処理を起動し、処理結果としての構造化文書又は文書部品を含むファイルをネットワーク経由で返信する。第1の構造化文書処理サーバは、該ファイルを第2の構造化文書処理サーバから受信して、第1の原料文書及び／又は第1の雛型文書として使用することができる。

## 【 0 0 5 5 】

また、第2の処理起動記述を入力する第2の構造化文書処理サーバは、前記ネットワーク経由の通信を必要としない第1の構造化文書処理サーバと同一のコンピュータ・システム上で構成されてもよい。この場合、第1の構造化文書処理サーバは、構造化文書又は文書部品を含むファイルに替えて、第2の構造化文書処理サーバによる処理結果である構造化文書又は文書部品を構文解析木として入力する切り替え手段を備えていてもよい。この結果、解析手段による重複した構文解析処理を省略することができ、処理速度が向上するとともに計算機負荷や通信負荷が軽減される。

## 【 0 0 5 6 】

また、本発明の第2の側面に係る構造化文書処理システムは、さらに、前記ファイル・サーバから入力した原料文書又は雛型文書の構文解析木をファイル名又は処理起動記述と対応付けて保持する保持手段と、ファイル名に対応する構造化文書ファイルを前記ファイル・サーバから入力する代わりに該当する構文解析木を前記保持手段から入力する入力手段とを備えていてもよい。この結果、保持手段にキャッシュされた構文解析木を有効に活用して、処理速度を向上させることができる。

# d

また、本発明の第1の側面に係る構造化文書処理システムも、文書部品の抽出を指定した抽出命令が付与された第1の構造化文書のファイル名と反復複写命令又は属性置換命令を付与した第2の構造化文書のファイル名を含む処理起動記述を入力として用いることができる。この場合、属性置換手段は、処理起動記述の一部と、文書部品にあらかじめ設定されている属性文字列と、置き換えてえられる文字列を、該文書部品の属性文字列として設定するようにしてもよい。

## 【 0 0 5 7 】



また、本発明の第 3 の側面は、構造的に表現され 1 以上の文書部品を含んだ構造化文書进行处理するための構造化文書処理システムであって、

構造化文書の構造を解析して構文解析木を生成する解析手段と、

前記解析手段により生成された構文解析木を基に構造化文書に付与された命令と文書部品を分離して命令を取り出すとともに、命令の文法的なエラーを検出してエラー情報を出力する命令分離手段と、

エラー情報を入力してエラー通知を行なう文書を合成するエラー通知文書合成手段と、

エラー通知文書へのアクセス情報を生成する処理起動記述合成手段と、

処理起動記述を解釈して、エラー通知文書を取り出す処理起動記述解析手段と

エラー通知文書を保持する保持手段と、

を具備することを特徴とする構造化文書処理システムである。

#### 【 0 0 5 8 】

ここで言うエラー情報は、エラーの種類やエラーの検出された文書中の位置（行数）など、通常のエラー処理でよく扱われるものでよい。

#### 【 0 0 5 9 】

エラー通知文書合成手段は、エラー情報を入力して、エラー通知を行なう文書を合成する。エラー情報は例えばエラー番号とし、エラー文書はエラー番号をファイル名とするファイルとする。エラー通知文書合成手段は、命令解析手段から入力されたエラー番号をファイル名とみなして、ファイルからエラー通知文書の 1 つを入力する。入力された文書は、解析手段を用いて構文解析木が作られる。構文解析木は保持手段に保持されるが、必ずしも構文解析木を作る必要はなく、文字列情報のまま保持手段に保持してもよい。

#### 【 0 0 6 0 】

処理起動記述合成手段が生成するエラー通知文書へのアクセス情報は、例えば、エラー通知ページにアクセスするための URL として記述される。エラー通知文書は、処理起動記述合成手段で合成された URL をキー情報として、保持手段に格納される。

## 【 0 0 6 1 】

本発明の第3の側面によれば、文書処理の途中で発生したエラーを、文書処理の合成結果とは別にして利用者に通知することができる。したがって、高度で柔軟な文書処理アプリケーションを構成することができる。さらに、エラーが発生した時点でエラー通知文書を生成して、文書処理の結果を一時的に格納する保持手段に（エラー通知文書のリクエストに先行して）格納し、後のエラー通知文書の要求に対して出力するように動作することができる。このため、エラーが発生したプロセスとエラーの種類やエラーに関する情報を対応付けて保持するような機構を用意する必要がなくなる。

## 【 0 0 6 2 】

また、本発明の第4の側面は、構造的に表現され1以上の文書部品を含んだ構造化文書进行处理するための構造化文書処理方法であって、

命令が付与された構造化文書を解析して構文解析木を生成するステップと、

構文解析木を探索して、文書部品と命令とを分離して命令を取り出すステップと、

構造化文書から取り出された各命令を併合・整列して、構造化文書进行处理するための命令列からなる文書処理記述を生成するステップ、

文書処理記述を解釈して、構造化文書を合成処理するステップと、  
を具備することを特徴とする構造化文書処理方法である。

## 【 0 0 6 3 】

また、本発明の第5の側面は、構造的に表現され1以上の文書部品を含んだ構造化文書进行处理するための構造化文書処理方法であって、

(a) 文書部品の抽出を指定する抽出命令が付与された第1の構造化文書を解析して構文解析木を生成するステップと、

(b) 文書部品の所定回数の複写を指定する反復複写命令又は文書部品の挿入若しくは置換を指定する挿入置換命令が付与された第2の構造化文書を解析して構文解析木を生成するステップと、

(c) 構文解析木を探索して、文書部品と命令とを分離して命令を取り出すステップと、

(d) 抽出命令が付与された第 1 の構造化文書から取り出された抽出命令と、反復複写命令及び／又は挿入置換命令が付与された第 2 の構造化文書から取り出された反復複写命令及び／又は挿入置換命令とを併合・整列して、該第 1 及び第 2 の構造化文書进行处理するための命令列からなる文書処理記述を生成するステップ

(e) 文書処理記述を解釈して、抽出命令の対象である第 1 の文書部品を第 1 の構造化文書から取り出すステップと、

(f) 文書処理記述を解釈して、反復複写命令の対象である文書部品と該文書部品付与された命令を所定回数だけ反復複写するステップと、

(g) 文書処理記述を解釈して、第 1 の文書部品を、挿入置換命令の対象である第 2 の文書部品の前又は後に挿入するか又は該第 2 の文書部品と置換するステップと、

(h) 前記ステップ (e) 乃至 (g) の結果得られた構文解析木を出力するステップと、

を具備することを特徴とする構造化文書処理方法である。

【0064】

【作用】

従来の構造化文書処理方式では、原料文書や雛形文書と別に文書进行处理するためのスクリプトを作成し、これを管理することが問題の原因であった。

【0065】

これに対し、本発明では、スクリプトのように手続きの体裁を備えた命令を用いずに構造化文書の処理を行うものである。

【0066】

図 1 には、本発明に係る構造化文書処理システムの基本概念を図解している。同図に示すように、この構造化文書処理システムは、命令付与手段と、解析手段と、命令分離手段と、文書処理記述合成手段と、インタプリタと、抽出手段と、反復複写手段と、挿入置換手段とで構成される。これら各手段は、例えば、計算機システム上の処理プロセスとして実現することができる。

【0067】

命令付与手段は、原料文書としての構造化文書内に各命令を埋め込む。ここで言う命令には、構造化文書の文書部品の取り出しを指定する抽出命令と、文書部品を所定の回数の反復複写の指定をする反復複写命令と、文書部品の挿入又は置換を指定する挿入置換命令が含まれる。

## 【0068】

これら各命令はそれ自体では、スクリプトのように処理を記述した「手続き」としての体裁を整えていない。例えば、構造化文書がHTMLやXMLなどのマークアップ言語で記述されている場合には、いわゆるコメント文の形式で元の文書内に埋め込まれる。

## 【0069】

解析手段は、各命令が埋め込まれた第1の構造化文書と第2の構造化文書とを受け取って、これらを解析する。

## 【0070】

命令分離手段は、入力された第1及び第2の構造化文書を文書部品と命令に分離して、各命令を取り出す。

## 【0071】

文書処理記述合成手段は、取り出した各命令を併合し、所定の順に並べて文書処理記述を合成する。

## 【0072】

インタプリタは、一般に、前記文書処理記述に含まれる命令列を順に解釈実行する。本発明においては、インタプリタは、抽出手段に文書部品を指定して、原料文書から文書部品を取り出させる。また、インタプリタは、反復複写手段に文書部品を指定して、文書部品や各命令を反復複写せしめる。さらに、挿入置換手段に2つの文書部品を指定して、文書部品の挿入又は置換を行なわしめる。この結果、文書部品同士を合成するという構造化文書の処理が実現する。

## 【0073】

本発明に係る構造化文書処理システムに入力されたそれぞれの構造化文書中には、文書部品を取り出す抽出命令と、反復複写と挿入置換命令が付与されている。したがって、文書部品の取り出しの指定と、反復複写の指定と、文書部品を挿

入または置換する文書部品（個所）の指定を行ない、入力した複数の構造化文書から取り出した命令を動的に合成して文書処理記述を作成するようにした結果、文書処理記述スクリプトが不要となる。したがって、元の文書と別にスクリプトを管理するという手間が省略される。

## 【 0 0 7 4 】

本発明によれば、文書処理記述は、入力文書の構造に合わせた抽出命令（列）と、反復複写命令（列）や挿入置換命令（列）とから合成される。本発明に係る構造化文書処理システムに構造化文書が入力されると、命令（列）が文書から分離され併合され、文書进行处理する前に文書処理記述が動的に合成される。したがって、合成された文書処理記述は、複数の入力文書の処理に適合した処理記述となっている。

## 【 0 0 7 5 】

本発明によれば、文書処理の度に入力文書に合わせた処理記述が動的に構成される。したがって、従来は必要であった文書処理スクリプトが不要となり、必然的に、文書と別にスクリプトを管理するという労力からも解放される。

## 【 0 0 7 6 】

本発明によれば、スクリプトを開発するためのプログラミングが不要になる。また、構造的に異なる文書が入力される毎に異なるスクリプトを用意する必要がない。利用者は、これらのスクリプトを入力文書に合わせて選択する必要がない。

## 【 0 0 7 7 】

また、入力文書の構造の変更に伴うスクリプトの変更の必要がないので、変更を必要とするスクリプトを探したり、変更し忘れによる不具合が発生しない。したがって、構造化文書の利用者にとっては、それぞれの目的に合わせた構造化文書処理のアプリケーションを作成して、出力結果を所望の目的に合致した形に簡単にカスタマイズできるシステムの実現が容易になる。すなわち、効率のよいアプリケーションの開発が簡単になる。

## 【 0 0 7 8 】

総じて、本発明によれば、構造化文書処理システムの開発や維持管理や利用が

簡単である。

【0079】

本発明のさらに他の目的、特徴や利点は、後述する本発明の実施例や添付する図面に基づくより詳細な説明によって明らかになるであろう。

【0080】

【発明の実施の形態】

以下では、説明の便宜上、構造化文書（以下、単に「文書」とする）としてHTML（Hyper Text Markup Language）言語で記述されたHTMLファイルであるとする。但し、XML（eXtensible Markup Language）やその他の言語で記述された構造化文書に対して本発明を適用できることは言うまでもない。

【0081】

図2には、本発明の第1の実施例に係る構造化文書処理システムの構成を模式的に示している。以下、同図を参照しながら、各ブロックについて説明する。

【0082】

命令付与手段は、入力される文書内に、抽出命令、反復複写命令、挿入置換命令（以下では、これらをまとめて単に「各命令」とする）を埋め込む。各命令は、次のように記述される。すなわち、

【0083】

【数5】

・抽出命令

<!--#OUT LABEL-->

※HTMLで記述された文書部品

<!--#/OUT-->

【0084】

【数6】

・反復複写命令

<!--#RP LABEL-->

※HTMLで記述された文書部品

<!--/#RP-->

【0085】

【数7】

・挿入置換命令

<!--#IN LABEL-->

※HTMLで記述された文書部品

<!--#/IN-->

【0086】

この例では、'OUT' が抽出命令を、'RP' が反復複写命令を、'IN' が挿入置換命令を、それぞれ表すものとする。また、LABELは任意の文字列でよい。<!--#OUT LABEL-->を抽出命令開始タグ、<!--#/OUT-->を抽出命令終了タグと呼ぶ。同様に、<!--#RP LABEL-->及び<!--#/RP-->を反復複写命令開始タグ及び反復複写命令終了タグとし、<!--#IN LABEL-->及び<!--#/IN-->を挿入置換命令開始タグ及び挿入置換命令終了タグとする。挿入置換命令開始タグと挿入置換命令終了タグの間には、何も書かなくてもよい。その場合には、文書合成処理以前には正しいHTMLファイルにならない場合もある。

【0087】

本実施例では、抽出命令を付与した構造化文書を原料文書とする。第1の原料文書のHTMLファイルを以下に示す。

【0088】

【数8】

<HTML>

<HEAD>

<TITLE> Sample1 </TITLE>

</HEAD>

<BODY>

<FONT SIZE="5"><!--#OUT グループ-->Aチーム<!--#/OUT-->10月度報告</FONT>

```

<H2>概要</H2>
<HR>
<!--#OUT サマリー-->
<UL>
  <LI> 項目1の進捗60% 計画どおり
  <LI> 項目2の進捗30% 遅れ: 3日
  <LI> 項目3の進捗70% 予定外の問題発生、計画見直しが必要
</UL>
<!--#OUT-->
<HR>
</BODY>
</HTML>

```

【0089】

第1の原料文書を解析木の形式で表現すると図3のようになる。また、この第1の原料文書を標準的なHTMLブラウザに表示した結果は、図18に示した通りとなる。該原料文書に埋め込まれた抽出命令”OUT”は、このうち破線で囲まれた部分の取り出しを要求するものである。

【0090】

また、第2の原料文書のHTMLファイルを以下に示す。

【0091】

【数9】

```

<HTML>
<HEAD>
<TITLE> Sample2 </TITLE>
</HEAD>
<BODY>
<H1>月度報告<!--#OUT グループ-->Bチーム<!--#OUT
T--></H1>
<H2>概要</H2>

```



```

<HR>
<!--#OUT サマリー-->
  <TABLE BORDER="0" CELSPACING="0"
  CELLPADDING="1">
    <TR><TH>項目</TH><TH>進捗</TH><TH AL
    IGN=LEFT">評価</TH></TR>
    <TR><TH>1</TH><TD>20%</TD><TD>やや
    計画おくれ(1日)</TD></TR>
    <TR><TH>2</TH><TD>40%</TD><TD>遅れ
    なし</TD></TR>
    <TR><TH>3</TH><TD>60%</TD><TD>進み
    : 3日</TD></TR>
  </TABLE>
<!--#OUT-->
<HR>
</BODY>
</HTML>

```

【0092】

第2の原料文書を解析木の形式で表現すると図4のようになる。また、この第2の原料文書を標準的なHTMLブラウザに表示した結果は、図19に示した通りとなる。該原料文書に埋め込まれた抽出命令"OUT"は、このうち破線で囲まれた部分の取り出しを要求するものである。

【0093】

また、本実施例では、反復複写命令"RP"と挿入置換命令"IN"を付与した構造化文書を雛型文書とする。本実施例において使用する第1の雛型文書のHTMLファイルを以下に示しておく。

【0094】

【数10】

```

<HTML>

```

```

<HEAD>
<TITLE> Template1 </TITLE>
</HEAD>
<BODY BGCOLOR=" #404040" >
    <H1>進捗サマリ</H1>
    <TABLE BORDER=" 1" >
        <TR><TH>チーム</TH><TH>概況</TH></TR>
<!--#RP グループ-->
    <TR>
        <TH>
<!--#IN グループ-->
        ここにグループ名が置換挿入される
<!--#/IN-->
        </TH>
        <TD>
<!--#IN サマリー-->
        ここに各グループの概況が置換挿入される
<!--#/IN-->
        </TD>
    </TR>
<!--#/RP-->
</TABLE>
</BODY>
</HTML>

```

【0095】

第1の雛型文書を解析木の形式で表現すると図5のようになる。

【0096】

また、本実施例において使用する第2の雛型文書のHTMLファイルを以下に示しておく。

【0097】

【数11】

```
<HTML>
<HEAD>
<TITLE> Template 2</TITLE>
</HEAD>
<BODY>
  <H1>進捗サマリ</H1>
  <TABLE BORDER=" 1" >
    <TR>
      <TH>チーム</TH>
<!--#RP グループ-->
      <TH>
<!--#IN グループ-->
        ここにグループ名が置換挿入される
<!--#/IN-->
      </TH>
<!--#/RP-->
    </TR>
    <TR>
      <TH>概況</TH>
<!--#RP サマリ-->
      <TD ALIGN=" LEFT" >
<!--#IN サマリ-->
        ここに各グループの概況が置換挿入される
<!--#/IN-->
      </TD>
<!--#/RP-->
    </TR>
```

```
</TABLE>
</BODY>
</HTML>
```

【0098】

第2の雛型文書を解析木の形式で表現すると図6のようになる。

【0099】

雛型文書において、LABELは、抽出命令と、反復複写命令または挿入置換命令の対応を指定するのに用いる。

【0100】

特定のLABEL引数が指定された抽出命令の数によって、同じラベルを有する反復複写命令の反復回数を定める。[数10]で示した第1の雛型文書の場合には、原料文書におけるLABEL引数「グループ」のみ反復複写される。これに対し、[数11]で示した第2の雛型文書の場合には、原料文書におけるLABEL引数「グループ」及び「サマリー」の双方が反復複写される。

【0101】

また、特定のLABELが指定された抽出命令によって抽出された文書部品は、同じLABEL引数を有する挿入置換命令によって挿入または置換挿入される。[数10]及び[数11]で示した各雛型文書の場合には、原料文書におけるLABEL引数「グループ」及び「サマリー」の双方に関して挿入又は置換挿入される。

【0102】

また、すべてのLABEL文字列が同一であるときは、LABEL文字列を省略したのと同等の効果をもたらす。

【0103】

本実施例において、各命令は、HTMLファイルの注釈（コメント文）として構成される。したがって、HTMLブラウザを始めとするHTMLを取り扱う標準的なアプリケーション上では、これら各命令は無視されるので、文書の内容を表示したり処理するアプリケーションに対して実質的に影響を及ぼさない。各命令は、後続の命令分離手段によって各HTMLで記述された文書から分離され、

取り出される。

【0104】

命令付与手段は、本実施例では、HTMLファイルを編集可能なエディタとして構成される。各命令は、単にHTMLファイル中に挿入された注釈タグなので、通常のテキスト・エディタあるいはHTMLエディタ上の編集作業を介して容易に付与することができる。例えば、ユーザは、テキスト・エディタを用いて、HTMLファイル中で取り出したい文書部品の前後それぞれに、抽出命令開始に相当する注釈タグ<!--#OUT LABEL-->、及び、抽出命令終了に相当する注釈タグ<!--#OUT-->をマニュアルで挿入することによって、命令を付与することができる。以下、<!--#OUT LABEL-->を抽出命令開始タグと呼び、<!--#OUT-->を抽出命令終了タグと呼ぶことにする。これら挿入される命令自体は、手続きの体裁をなしていないという点を充分理解されたい。なお、ここで用いられるテキスト・エディタは、一般的なエディタでよく、本明細書中ではこれ以上説明しない。

【0105】

また、別の実現方法としては、計算機システム上で文書の構造を自動的に解析して、特定のマークアップで挟まれた文書部品の前後に、同様な各命令を挿入することも可能である。すなわち、特開平6-52161号公報（前述）において開示された方式を用いて構造を指定した検索を行ない、所望の文書部品の前後に所望の各命令の開始タグと終了タグを文書に流し込み合成を行なうことによって、文書に対して命令を付与することができる。また、通常のテキスト処理によっても可能である。以下にperl言語のスクリプトで記述した例を示す（perlは米Net Labs社のLarry Wall氏が開発したインタプリタ言語であり、主としてUNIX系のOS上で使用される）。

【0106】

【数12】

```
$htmlfile = "sample.html";
open(HTML, "$htmlfile") || die "Can't
open $htmlfile."
```

```

while (<HTML>) {
    s/<TABLE [^>] *>/<!--#OUT LABEL-->$&
/gi;
    s/<¥/TABLE [^>] *>/$&<!--#¥/OUT-->/g
i;
    print;
}

```

## 【0107】

上記の例では、正規表現によるパターン・マッチを用いて文書部品の検索を行なう。前記スクリプトは、HTMLファイルを入力して、<TABLE>タグを検索してその直前に<!--#OUT LABEL-->を挿入するとともに、</TABLE>タグを検索してその直後に<!--#/OUT-->を挿入して、結果を出力する。

## 【0108】

次いで、入力手段と出力手段について説明する。

## 【0109】

入力手段は、ファイル名を指定して、ファイル・サーバからHTMLファイルを入力する。ファイル・サーバは例えばWWW (World Wide Web) サーバである。WWW空間上では、URL (Uniform Resource Locator) をHTMLファイルの名前としてアクセスすることができる。

## 【0110】

入力手段は、ネットワークに接続されたWWWサーバにネットワーク・コネクションを構成しており、HTTP (HyperText Transfer Protocol) として知られるプロトコルでHTMLファイルを転送する。すなわち入力手段は、HTMLファイル名を含むHTTPリクエストをWWWサーバに送信するとともに、WWWサーバからはHTMLファイルを含むHTTPレスポンスを受信する。より具体的には、所定のポート番号で待ち受けるWWWサ

サーバに対して、TCP/IP (Transmission Control Protocol/Internet Protocol) コネクションを形成し、HTTPのリクエストを入力する。HTTPリクエストには、URLとして与えられたファイル識別子が含まれている。これに対し、WWWサーバは、URLに基づいて、HTMLファイルをローカルなファイル・システムから読み出すか、所定のプログラムを起動してHTMLファイルを合成するか、又は、他のWWWサーバからHTMLファイルを入力するかして、HTTPリクエストで指定されたURLに対するHTMLファイルを用意し、これをHTTPレスポンスに含めて入力手段に出力する。

## 【0111】

このような入力手段の実装は、例えば米サンマイクロシステムズ社のJDK (Java Development Kit) に含まれる、java.net.URL、及び、java.net.URLConnectionクラスの実装を用いればよい。

## 【0112】

出力手段は、構造化文書をHTML形式に変換したものを、HTTPレスポンスの本体すなわちbody部として、ネットワークを経由して出力する。すなわち、本実施例の構造化文書処理システムは、クライアントからネットワークを経由して適当なリクエストを受信し、ファイル・サーバから原料文書と雛型文書を入力して処理を行ない、処理結果をレスポンスとしてクライアントに出力する構造化文書処理サーバとして実現される。特に、本実施例に係る構造化文書処理サーバは、HTTPリクエストを入力して所定の構造化文書の処理を行ない、HTTPレスポンスを出力するHTTPサーバとして実現する。

## 【0113】

出力手段に送られる構造化文書は、該構造化文書の構文解析木とする。構文解析木については後述する。構文解析木をHTMLに変換するには、該構文解析木を深さ優先順に走査して、開始タグ、テキストコンテンツ、及び、終了タグを出力する。走査において訪れたノードがテキスト・コンテンツを保持するノードである場合には、テキスト文字列を出力する。そうでなければ、開始タグ（例えば

TABLEタグなら”<TABLE>”)を出力し、次に子ノードを順に走査し、各子ノードについて再帰的に開始タグと終了タグとテキスト・コンテンツをHTMLとして出力する。子ノードの出力の後に、該ノードを上向きに訪れるときに終了タグ（例えばTABLEタグなら”</TABLE>”)を出力する。HTMLをHTTPレスポンスに含めて出力することは、通常のHTTPサーバの場合と同様なので、本明細書中ではこれ以上説明しない。また、HTTPリクエストを入力する部分については、本発明の要旨とは直接関係しないので、説明を省略する。このようなHTTPサーバの実現例としては、Apache WWWサーバがよく知られている。

## 【0114】

次いで、解析手段と分離手段について説明する。

## 【0115】

解析手段は、テキスト形式で記述されたHTMLファイルを入力し、HTMLの文法（構文）に従って解析し、構文解析木を構成するHTMLパーザである。本発明を実現する上で、従来技術と同様のHTMLパーザを利用することができるので、本明細書中では詳しく説明しない。（HTMLパーザの実現例については、Corporation for National Research Initiatives, 「Python Library Reference Release 1.5.2」12.2節 `html lib - A parser for HTML documents`などに記載されている。また、構文解析の方式については、A. V. エイホ他、コンパイラサイエンス社4章に詳述されている）。

## 【0116】

本実施例で用いるHTMLパーザは、テキスト形式のHTMLファイルを入力すると、木構造のデータを出力する。木構造の各ノードはメモリに一時格納される。各ノードは、HTMLタグ名または‘TEXT’をノード名として属性に持つ。この場合、各ノードがHTMLタグのマークアップ又はテキスト・コンテンツに対応する。また、HTMLタグの属性（Attribute）を属性に持つ。さらに子ノードへのポインタの配列を持つ。この配列は、第1の要素、第2の



要素…の順に、第2子ノード、第2子ノード…へのポインタが格納される。解析木中の任意のノードを指定することで、該ノード以下のサブツリーが文書部品として定まる。（木構造の実現については、例えば「アルゴリズムとデータ構造」岩波書店、p. 48～p. 54に詳解されている。）

## 【0117】

第1の原料文書、第2の原料文書、第1の雛型文書、第2の雛型文書の構文解析木（以下、「文書解析木」と呼ぶ）は、それぞれ図3、図4、図5、図6に示してある通りである。

## 【0118】

次いで、解析手段としてのHTMLパーザは、構文解析木を行きがけ順（prefix order）に巡回し、ノード名が注釈タグであって、注釈タグの中に記述されている注釈文字列が、次の規則のいずれかに該当するか否かを、文字列のパターン・マッチによって調べる。但し、「→」記号の左辺がパターンであり、右辺が各命令のいずれかである。また、＜空白＞は1以上の空白記号を意味し、＜LABEL文字列＞は、空白を含まない任意の文字列を意味するものとする（木構造を行きがけ順に巡回する方法については、例えば「アルゴリズムとデータ構造」岩波書店、p. 52～p. 53に木構造の前順走査として記載されている）。

## 【0119】

## 【数13】

" #OUT<空白><LABEL文字列>"	→	抽出命令開始タグ
" #/OUT<空白><LABEL文字列>"	→	抽出命令終了タグ
" #RP<空白><LABEL文字列>"	→	反復複写命令開始タグ
" #/RP<空白><LABEL文字列>"	→	反復複写命令終了タグ
" #IN<空白><LABEL文字列>"	→	挿入置換命令開始タグ
" #/IN<空白><LABEL文字列>"	→	挿入置換命令終了タグ

## 【0120】

このようなパターン・マッチは、正規表現による文字列マッチを用いて実現可能である。Perl言語で利用可能なパターン・マッチによって各命令の開始タグと終了タグを抽出するプログラムの例を、以下に示しておく。

【0121】

【数14】

```
$htmlfile = "Template2.html";
open(HTML, "$htmlfile") || die "Can't
open $htmlfile.";
while(<HTML>) {
    @__=split('>', $__);
    foreach $__ (@__) {
        $value='';
        $value="抽出命令開始タグ/" . $1 if /<!--
#OUT_([^\s]+)\s*--/;
        $value="反復複写命令開始タグ/" . $1 if /<!--
#RP_([^\s]+)\s*--/;
        $value="挿入置換命令開始タグ/" . $1 if /<!--
#IN_([^\s]+)\s*--/;
        $value="抽出命令終了タグ/" . $1 if /
<!--#%/OUT_([^\s]+)\s*--/;
        $value="反復複写命令終了タグ/" . $1 if /<!--
#%/RP_([^\s]+)\s*--/;
        $value="挿入置換命令終了タグ/" . $1 if /<!--
#%/IN_([^\s]+)\s*--/;
        print "$value\n" if $value ne '';
    }
}
```

【0122】

文書解析木中のアドレスは、文書部品のノードの位置を解析木のルート・ノードからのパスを示したベクトルによって表される。ベクトルの各要素はパス上に存在するノードに対応する。深さが3の位置にある文書部品のアドレスは、3次元のベクトルで表現され、ベクトルの各要素は、それぞれのノードの親に対して（長子を0と数えて、）何番目の子であるかを表す数とする。例えば、[0 1 2] というアドレス表現は、ルートの第2子の第3子に位置するノード以下のサブツリーで表される文書部品のアドレスである。所望のノードのアドレスが指定されれば、次のステップではルート・ノードから該所望のノードを対象ノードとする。

【0 1 2 3】

【数 1 5】

Step 1: 対象ノードをルート・ノードに設定し、「深さ」を1に設定する。

Step 2～Step 4をベクトルの次元数-1だけ繰り返す

Step 2: アドレス・ベクトルの各要素のうち「深さ」番目の番号（iとする）を取り出す。（ただし先頭を0番目とする）

Step 3: 対象ノードが保持する子ノード配列から「i」番目を取り出し、新たに対象ノードとする（ただし配列の先頭を0番目とする）。

Step 4: 「深さ」に1を加える。

【0 1 2 4】

このような解析木のアドレス表現について、図5を参照しながら説明する。同図においては、反復複写命令 [0 1 1 1] が付与された文書部品は、[0 1 1 2] のノード以下のサブツリーであり、第1の挿入置換命令 [0 1 1 2 0 0] が付与された文書部品は、[0 1 1 2 0 1] のノード以下のサブツリーであり、第2の挿入置換命令 [0 1 1 2 1 0] が付与された文書部品は、[0 1 1 2 1 1] のノード以下のサブツリーである。

【0 1 2 5】

文書部品IDは、文書部品を識別するためのもので、このIDを指定して文書部品を取り出すことができる。本実施例では、文書解析木IDと文書部品のアド

レスの組で文書部品 I D を表すものとする。文書解析木の I D は、例えば、構成した解析木の順に割り振った番号としてもよいし、文書解析木を格納したメモリ上のアドレスとして、例えば文書解析木のルート・ノードへのポインタでもよい。

## 【 0 1 2 6 】

次いで、命令分離手段について説明する。本実施例の命令分離手段は、文書解析木を行きがけ順に巡回し、各命令に対するオペコードと、ラベル文字列と、各命令が付与された文書部品の I D を前記オペコードに対するオペランドとする 3 項目からなる命令列を出力する。オペコードは、抽出命令は O U T、反復複写命令は R P、挿入置換命令は I N とする。文書解析木の I D は、第 1 の原料文書を 1、第 2 の原料文書を 2、第 1 の雛型文書を 3、第 2 の雛型文書を 4 とする。それぞれの文書から分離した命令列を以下に示す。

## 【 0 1 2 7 】

## 【数 1 6】

第 1 の原料文書から分離した命令列

オペコード	ラベル	文書部品 I D
O U T	グループ	< 0, [ 0 1 0 1 ] >
O U T	サマリ	< 0, [ 0 1 4 ] >

## 【 0 1 2 8 】

## 【数 1 7】

第 2 の原料文書から分離した命令列

オペコード	ラベル	文書部品 I D
O U T	グループ	< 1, [ 0 1 0 2 ] >
O U T	サマリ	< 1, [ 0 1 4 ] >

## 【 0 1 2 9 】

## 【数 1 8】

第 1 の雛型文書から分離した命令列

オペコード	ラベル	文書部品 I D
R P	グループ	< 2, [ 0 1 1 2 ] >

IN           グループ           <2,    [0  1  1  2  0  1]>  
 IN           サマリ            <2,    [0  1  1  2  1  1]>

【0130】

【数19】

第2の雛型文書から分離した命令列

オペコード	ラベル	文書部品ID
RP	グループ	<3,    [0  1  1  0  1]>, <
		3,    [0  1  1  0  2]>
IN	グループ	<3,    [0  1  1  0  2  1]>
RP	サマリ	<3,    [0  1  1  1  1]>, <
		3,    [0  1  1  1  2]>
IN	サマリ	<3,    [0  1  1  1  2  1]>

【0131】

次いで、文書処理記述合成手段について説明する。文書処理記述合成手段は、複数の原料文書または雛型文書から分離した命令列を入力し、命令列を併合し、順序の変更をし、変換と合成を行なって文書処理記述を出力する。

【0132】

ここで、原料文書1と原料文書2と雛型文書1のそれぞれから分離した命令列から文書処理記述を構成する処理を例にとって、文書処理記述合成手段の動作について説明する。文書処理記述合成手段は、以下に示す処理A～処理Cを順に行なう。

【0133】

#### ■処理A： OUTの計数

原料文書1と原料文書2のそれぞれから分離した命令列に含まれるOUTの個数をラベル文字列毎に数える。これは、以下に示すように、命令列を走査して、オペコードがOUTである命令について、ラベル文字列をキーとして、整数値を値とする連想配列をインクリメントすることで実現できる。走査が終了した後で、ラベル文字列をキーとしてOUT命令の個数を各ラベル文字列ごとに取り出すことができる。

【 0 1 3 4 】

【 数 2 0 】

すべて原料文書のすべての命令列について

Step 1 1 : オペコードが OUT なら Step 1 2 を実行。そうでなければ次の命令を調べる

Step 1 2 : オペランドのラベル文字列をキーとして、連想配列を調べる。

連想配列が未登録であれば、値として 1 を設定する。

連想配列が登録されていれば、値に 1 を加える。

Step 1 3 : 次の命令について Step 1 1 を実施。

【 0 1 3 5 】

■ 処理 B : RP による各命令の複写

雛型文書 1 の命令列において、命令列中に RP が含まれていた場合、すべての RP のオペランドの文書部品中に含まれる IN と OUT と RP を、原料文書の OUT 命令の数に従って複写する。この複写は 1 つの雛型文書について、以下に示す手順 Step 2 1 ~ Step 2 6 を実行することで行われる。

【 0 1 3 6 】

【 数 2 1 】

Step 2 1 : RP リストを作成する。

Step 2 2 : RP リストから、対象とする RP (反復複写命令) を順に変更しつつ各 RP について Step 2 3 以降の処理を実行する。

Step 2 3 : 対象 RP のオペランドのラベル文字列を取り出し、上記の処理 A で作成した連想配列から値を取り出し、反復複写回数  $t$  とする。 $t$  が 2 以上なら Step 2 4 以降の手順を実行する。

Step 2 4 : 反復複写回数  $t$  を RP 命令のオペランドのラベル文字列に代えて、文書処理記述に出力する。

Step 2 5 : オペランドから、文書部品の ID を取り出して、第 1 のアドレスを取り出す。第 1 のアドレスの次元数 (要素の数) を  $k$  とする。

Step 2 6 : 命令列の各命令のうち、Step 2 2 で取り出した第 1 のアドレスより次元数が大きく、かつ、先頭  $k$  個が一致するアドレスをオペランドに持

つ命令を取り出す。

例えば、 $[0 \ 1 \ 2 \ 3 \ 4]$ と $[0 \ 1 \ 2]$ は、先頭3個が一致するので、 $[0 \ 1 \ 2 \ 3 \ 4]$ は $[0 \ 1 \ 2]$ のノードの子孫である。また、 $[0 \ 1 \ 1 \ 0 \ 2]$ と $[0 \ 1]$ は先頭2個が一致する。 $[0 \ 1 \ 1 \ 0 \ 2]$ は $[0 \ 1]$ のノードの子孫である。このようなチェックは、2つの配列間で要素同士を比較することで容易に実現できる。

Step 27: Step 24で取り出した命令を複写し、複写された命令のオペランドのアドレス（第2のアドレス）を次のように書き換える。

第1のアドレス =  $[A_1 \ A_2 \ A_3 \ \dots \ A_{k-1} \ A_k]$  (RP命令のオペランドの文書部品のアドレス)

第2のアドレス =  $[A_1 \ A_2 \ A_3 \ \dots \ A_{k-1} \ A_k \ N_1 \ N_2 \ N_3 \dots]$

書き換え後のアドレス =  $[A_1 \ A_2 \ A_3 \ \dots \ A_{k-1} \ A_{k+i} \ N_1 \ N_2 \ N_3 \dots]$

但し、 $i$ は、複写の度に増加する整数、すなわち、 $1, 2, \dots, t-1$ とする。また、RP命令のオペランドに複数の連続した文書部品が指定されている場合は、指定されている文書部品の数を $n$ 個とすると、 $i$ は、複写の度に $n$ ずつ増加する。例えば $n=2$ の場合、 $i$ は、 $2, 4, \dots, (t-1) \times n$ とする。

【0137】

複写された命令以外のすべての命令について、オペランドの文書部品のアドレス（第3のアドレス）の先頭 $k-1$ 個が一致し、かつ、 $k$ 番目が $A_k$ より大きいものについて、アドレスの付け替えを行なう。

【0138】

【数22】

第1のアドレス =  $[A_1 \ A_2 \ A_3 \ \dots \ A_{k-1} \ A_k]$  (RP命令のオペランドの文書部品のアドレス)

第3のアドレス =  $[A_1 \ A_2 \ A_3 \ \dots \ A_{k-1} \ B \ N_1 \ N_2 \ N_3 \dots]$

書き換え後のアドレス = [ A1 A2 A3 … Ak-1 B+t  
-1 N1 N2 N3…]

但し、 $B > A_k$ とし複写する度に増加する整数、すなわち、1、2、…、 $t-1$ とする。

【0139】

この付け替えは、反復複写によるアドレスのずれを補正する処理となる。複写した命令がRPの場合は、Step 21で作成したRPリストに追加する。

【0140】

【数23】

第1の雛型文書から分離した命令列に処理Bを施した結果えられる命令列

オペコード	ラベル	文書部品ID
RP	2	<2, [0 1 1 2] >
IN	グループ	<2, [0 1 1 2 0 1] >
IN	サマリ	<2, [0 1 1 2 1 1] >
IN	グループ	<2, [0 1 1 3 0 1] >
IN	サマリ	<2, [0 1 1 3 1 1] >

【0141】

【数24】

第2の雛型文書から分離した命令列に処理Bを施した結果えられる命令列

オペコード	ラベル	文書部品ID
RP	2	<3, [0 1 1 0 1] >, <3, [0 1 1 0 2] >
IN	グループ	<3, [0 1 1 0 2 1] >
IN	グループ	<3, [0 1 1 0 4 1] >
RP	サマリ	<3, [0 1 1 1 1] >, <3, [0 1 1 1 2] >
IN	サマリ	<3, [0 1 1 1 2 1] >
IN	サマリ	<3, [0 1 1 1 4 1] >

【0142】



# ■処理C： 命令の合成

すべての原料文書のOUTと、雛型文書のINとを、ラベル文字列が一致するものどうしでペアを構成して各ペアから新しい命令を合成する。この動作手順について以下に説明する。

## 【0143】

本実施例では、例えば”OUT LABEL 文書部品1”と”IN LABEL 文書部品2”というペアから”FILL 文書部品1 文書部品2”を合成する。FILLは、文書部品1を文書部品2と置換する命令を表す。この動作によって原料文書に出現する文書部品の出現順にかかわらず、雛型文書の中のユーザが所望する位置に、前記原料文書の文書部品を置換または挿入することが可能になる。

## 【0144】

### 【数25】

Step 31： 原料文書のすべての命令列を走査し、ラベル文字列ごとに命令の配列を作成する。この配列はOUT配列と呼ぶ。OUT配列は、ラベル文字列をキーとする連想配列に格納しておく。

Step 32： 雛型文書のすべての命令列を走査し、ラベル文字列ごとに命令の配列を作成する。この配列はIN配列と呼ぶ。IN配列は、ラベル文字列を属性として保持する。

Step 33： すべてのIN配列について、Step 34以降を実行する。

Step 34： IN配列の属性であるラベル文字列から、一致する文字列をオペランドに持つOUT配列を取り出す。

Step 35： IN配列のすべての要素に対して、0番目から順にm番目を対象要素としてStep 36以降を実行する。

Step 36： OUT配列のm番目の要素として格納されたOUT命令のオペランドの文書部品のIDである文書部品1を取り出す。IN配列のm番目の要素として格納されたIN命令のオペランドの文書部品のIDである文書部品2を取り出す。

Step 37： 文書処理記述に次の命令を追加する。この例では、オペコードは

F I L L で、オペランドは文書部品 1 と文書部品 2 である

F I L L      文書部品 1      文書部品 2

【 0 1 4 5 】

また、文書部品に含まれるコンテンツ、例えば「社員番号」を取り出して、社員番号順に O U T 命令を整列して O U T 配列を構成し、雛型文書中の I N 命令の出現順に構成した I N 配列に対応付けて、F I L L を構成することも可能である。このように、原料文書の O U T 命令で抽出される文書部品と、I N 命令で指定される挿入置換の場所との対応付けは、ラベル文字列の対応付けや、コンテンツの内容による整列だけにとどまらず、さまざまな方法が実施可能である。例えば、テーブルを検索して、順序を決めることもできるし、どのようなサーバから得られた原料文書か（例えばホスト名）によって、挿入置換場所を変更することも可能である。この場合は、例えば I N 命令のラベル文字列をホスト名にしておき、O U T 命令のオペランドの文書 I D と、ホスト名の対応表を基に、同様の処理をすることで実現できる。

【 0 1 4 6 】

これらの処理によって文書処理記述が合成される。文書処理記述の例を以下に示しておく。

【 0 1 4 7 】

【数 2 6】

※第 1 の原料文書と第 2 の原料文書と第 1 の雛型文書から合成された文書処理記述例 1

R P	2	2, [ 0 1 1 2 ]
F I L L	0, [ 0 1 0 1 ]	2, [ 0 1 1 2 0 1 ]
F I L L	1, [ 0 1 0 2 ]	2, [ 0 1 1 3 0 1 ]
F I L L	0, [ 0 1 4 ]	2, [ 0 1 1 2 1 1 ]
F I L L	1, [ 0 1 4 ]	2, [ 0 1 1 3 1 1 ]

【 0 1 4 8 】

【数 2 7】

※第 1 の原料文書と第 2 の原料文書と第 2 の雛型文書から合成された文書処理記

## 述の例 2

```

RP      2      3, [0 1 1 0 1]      3, [0 1
  1 0 2]
RP      2      3, [0 1 1 1 1]      3, [0 1
  1 1 2]
FILL    0, [0 1 0 1]      3, [0 1 1 0 2 1]
FILL    1, [0 1 0 2]      3, [0 1 1 0 3 1]
FILL    0, [0 1 3]        3, [0 1 1 1 2 1]
FILL    1, [0 1 3]        3, [0 1 1 1 3 1]
      【0149】

```

上記の例では、1つのRP命令は、2つの連続する文書部品に付与されているので、RP命令のオペランドは2個指定されていることに注意する。また、FILL命令を合成するのではなく、文書部品を格納する一時領域を指定するラベルを用意することで、次のような文書合成記述を出力してもよい。

【0150】

【数28】

※原料文書1と原料文書2と雛型文書2から合成された文書処理記述の例3

```

RP      2      2, [0 1 1 2]
OUT     0, [0 1 0 1]      tmp
IN      tmp      2, [0 1 1 2 0 1]
OUT     0, [0 1 4]      tmp
IN      tmp      2, [0 1 1 2 1 1]
OUT     1, [0 1 0 2]      tmp
IN      tmp      2, [0 1 1 3 0 1]
OUT     1, [0 1 4]      tmp
IN      tmp      2, [0 1 1 3 1 1]

```

【0151】

次に、インタプリタと、抽出手段と、反復複写手段と、挿入置換手段の動作について説明する。

## 【 0 1 5 2 】

上述した文書処理記述がインタプリタに入力されると、インタプリタは、文書処理記述の先頭から順に走査し、オペコードを判定して、抽出手段、反復複写手段、又は挿入置換手段のいずれかに、オペランドで指定された文書部品 I D を入力する。O U T 命令には抽出手段が、R P 命令には反復複写手段が、I N 命令には挿入置換手段が、それぞれ対応するものとする。挿入置換手段には、文書部品の I D に加えて、挿入または置換する文書部品を入力する。F I L L の場合には、第 1 のオペランドで指定された文書部品 I D が抽出手段に入力され、その結果として得られた文書部品と、第 2 のオペランドで指定された文書部品 I D を挿入置換手段に入力する。

## 【 0 1 5 3 】

本実施例のインタプリタは、命令記述を読み込んで、各命令を 1 ステップ毎に順次解釈して所定の処理を実行する処理を行なうような通常のインタプリタでよい。かかるタイプのインタプリタの実現は当業界において既に周知なので、本明細書では説明を省略する（例えば、インタプリタの実現方法は、滝口政光、「作りながら学ぶコンパイラ／インタプリタ」（C Q 出版株式会社）に詳しく説明されている）。本実施例では、文書処理記述内の命令ステップの順に、各オペコードに対して所定の処理手段が選択され、各処理手段にオペランドとして文書部品の I D が入力される。

## 【 0 1 5 4 】

抽出手段は、入力された文書部品の I D から、指定された文書解析木のノードを見つけて、コピーした文書部品を返す。コピーは該ノード以下のすべてのノードを行きがけ順に巡回し、各ノードごとにコピーするものとする。コピーした文書部品は、メモリ上に格納されてポインタが返されるように実装してもよい。先の文書処理記述例 3 の一時格納領域として指定した t m p は、このようなコピーした文書部品を格納できるように実装してもよいし、前記ポインタであってもよい。

## 【 0 1 5 5 】

反復複写手段は、入力された文書部品の I D から、指定された文書解析木のノ

ードを見つけて、該文書解析木上で反復複写する。（反復複写した結果、アドレスがずれるノードがあるため、先の処理 B の Step 24 で、オペランドに指定された文書部品のアドレスの付け替えを行なっていることに注意する。）

## 【0156】

図 7 には、雛型文書 1 に対して、文書処理記述例 1 の RP 命令で反復複写した結果の文書解析木の例を示している。また、図 8 には、雛型文書 2 に対して、文書処理記述例 2 の RP 命令で反復複写した結果の文書解析木の例を示している。太枠で囲まれた部分が反復複写された部分を表している。

## 【0157】

挿入置換手段には、挿入又は置換する第 1 の文書部品（又はその ID）に加えて、挿入置換をする個所を示す第 2 の文書部品の ID を入力する。入力された第 2 の文書部品の ID から、指定された文書解析木のノードを見つけて、前（兄ノード）又は後ろ（弟ノード）に、指定した第 1 の文書部品のノードを挿入する。置換動作の場合は、第 2 の文書部品として指定されたノードを削除して、その代りに指定した別の文書部品のノードを挿入する。第 1 又は第 2 の文書部品の代りに複数の文書部品を指定して、挿入又は置換するように実現してもよい。抽出手段、反復複写手段、挿入置換手段の各々は、上記のように単に文書解析木のツリー構造の変形や変換を行なうものであり、その実現は当業界において既に周知なので、本発明ではこれ以上説明しない（木構造に対する要素の挿入や削除については、例えば「アルゴリズムとデータ構造」（岩波書店）、2. 4 節に説明されている）。

## 【0158】

インタプリタによって、文書処理記述が解釈され雛型文書から構成された解析木が処理され、文書が合成される。図 9 には、第 1 の原料文書と第 2 の原料文書と第 1 の雛型文書から合成した文書の解析木を示している。同図において、網掛けを付されたノードが、置換された文書部品に相当する。合成結果の HTML ファイルを以下に示しておく。

## 【0159】

## 【数 2 9】

```

<HTML>
<HEAD>
<TITLE> Template1 </TITLE>
</HEAD>
<BODY BGCOLOR=" #404040" >
    <H1>進捗サマリ</H1>
    <TABLE BORDER=" 1" >
        <TR><TH>チーム</TH><TH>概況</TH><
/TH>
<!--#RP グループ-->
    <TR>
        <TH>
<!--#IN グループ-->Aチーム<!--#/IN-->
        </TH>
        <TD>
<!--#IN サマリ--><UL>
            <LI> 項目1の進捗60% 計画どおり </LI>
            <LI> 項目2の進捗30% 遅れ: 3日 </
LI>
            <LI> 項目3の進捗70% 予定外の問題発生、
計画見直しが必要 </LI>
        </UL><!--#/IN-->
        </TD>
    </TR>
    <TR>
        <TH>
<!--#IN グループ-->Bチーム<!--#/IN-->
        </TH>

```

```

<TD>
<!--#IN サマリー--><TABLE BORDER="0" CELL
SPACING="0" CELLPADDING="1">
<TR><TH>項目</TH><TH>進捗</T
H><TH ALIGN="LEFT">評価</TH></TR>
<TR><TH>1</TH><TD>20%</T
D><TD>やや計画おくれ(1日)</TD></TR>
<TR><TH>2</TH><TD>40%</T
D><TD>遅れなし</TD></TR>
<TR><TH>3</TH><TD>60%</T
D><TD>進み: 3日</TD></TR>
</TABLE><!--# /IN-->
</TD>
</TR>
<!--# /RP-->
</TABLE>
</BODY>
</HTML>

```

【0160】

上記のHTMLファイルを標準的なHTMLブラウザで表示した結果を図20に示しておく。

【0161】

同様に、第1の原料文書と第2の原料文書と第2の雛型文書から合成した文書のHTMLファイルを以下に示す。

【0162】

【数30】

```

<HTML>
<HEAD>
<TITLE> Template2 </TITLE>

```

```

</HEAD>
  <BODY>
    <H1>進捗サマリ</H1>
    <TABLE BORDER=" 1" >
      <TR>
        <TH>チーム</TH>
      <!--#RP グループ-->
      <TH>
        <!--#IN グループ-->Aチーム<!--#/IN-->
        </TH>
        <TH>
        <!--#IN グループ-->Bチーム<!--#/IN-->
        </TH>
      <!--#/RP-->
    </TR>
    <TR>
      <TH>概況</TH>
    <!--#RP サマリ-->
      <TD ALIGN=" LEFT" >
        <!--#IN サマリ-->
          <UL>
            <LI> 項目1の進捗60% 計画どおり </LI>
            <LI> 項目2の進捗30% 遅れ: 3日 </LI>
            <LI> 項目3の進捗70% 予定外の問題発生、
            計画見直しが必要 </LI>
          </UL><!--#/IN-->
        </TD>

```



```

        <TD ALIGN=" LEFT" >
<!--#IN サマリー-->
        <TABLE BORDER=" 0" CELLSP
ACING=" 0" CELLPADDING=" 1" >
        <TR><TH>項目</TH><TH>進捗</T
H><TH ALIGN=" LEFT" " >評価</TH></TR>
        <TR><TH>1</TH><TD>20%</T
D><TD>やや計画おくれ(1日)</TD></TR>
        <TR><TH>2</TH><TD>40%</T
D><TD>遅れなし</TD></TR>
        <TR><TH>3</TH><TD>60%</T
D><TD>進み: 3日</TD></TR>
        </TABLE><!--#/IN-->
    </TD>
<!--#/RP-->
    </TR>
</TABLE>
</BODY>
</HTML>

```

【0163】

上記のHTMLファイルを標準的なHTMLブラウザで表示した結果を図21に示しておくが、これは図20に示した文書の合成結果とは明らかに相違する。すなわち、同じ原料文書1及び原料文書2に対して本実施例に係る構造化文書処理を実行した場合であっても、適用した雛型文書が相違すれば合成文書も異なるという点を理解されたい。

【0164】

以上説明したように、本実施例に係る構造化文書処理システムは、命令付与手段によって抽出命令が付与された第1の構造化文書と、同じく命令付与手段によって反復複写命令と挿入置換命令が付与された第2の構造化文書とを入力とし、

解析手段によって各文書毎に文書解析木を構成して、命令分離手段によって各文書から命令列を取り出し、取り出した複数の該命令列を文書処理記述合成手段によって併合、整列、及び変換を行なって文書処理記述を合成し、インタプリタによって文書処理記述を順に走査し、文書処理記述中に含まれる各命令に従って抽出手段によって第 1 の文書部品を第 1 の文書から抽出し、反復複写手段によって第 2 の構造化文書の文書部品を第 1 の文書部品の数に依存して定まる回数だけ反復複写し、挿入置換手段によって第 2 の構造化文書に第 1 の文書部品を挿入又は置換することで、構造化文書の合成を行なうことができる。

## 【 0 1 6 5 】

本実施例に係る構造化文書処理システムによれば、

1. 原料文書の構造を変更した際に、これを処理するための別に管理されるスクリプトを用意しなくてよい。
2. 構造が異なる複数の原料文書を混在させて文書部品を取り出すには、各文書毎に文書部品に抽出命令を付与することで行なうので、文書毎の抽出の処理を個別に指定しなくてよい。同様に、文書部品を挿入又は置換する処理の指定も不要である。また、ユーザは、文書の構造が変わる度に適切なスクリプトを指定する必要がない。
3. 文書部品は、所望の命令を元の文書中に直接的に付与することと、取り出した文書部品と挿入置換される文書部品の条件を指定すること（例えばラベル文字列が一致するなど）によって作成することができる。また、各ユーザがそれぞれの目的に応じて雛型文書をデザイン（作成）したり、所望の処理を行なう命令が組み込まれた雛型文書を選択して、ユーザ自らが原料文書と組み合わせることで文書処理アプリケーションを構成することができる。このように手続き的でない、言い換えれば宣言的な方法で文書処理を構成することができるので、プログラミングの知識を充分に持たないユーザが広く文書処理を行なうことができる。すなわち、各ユーザがそれぞれの目的に合わせた構造化文書処理のアプリケーションを容易に作成できる文書処理システムが実現できる。

## 【 0 1 6 6 】

次いで、本発明の第 2 の実施例について説明する。

## 【 0 1 6 7 】

図 1 0 には、本発明の第 2 の実施例に係る構造化文書処理システムの構成を模式的に示している。本実施例では、抽出命令に代えて属性抽出命令が、挿入置換命令に代えて属性置換命令が、それぞれ付与された構造化文書に対して文書処理を実行するものである。以下、第 1 の実施例との相違点を中心に本実施例に関して説明する。

## 【 0 1 6 8 】

命令付与手段は、第 1 の実施例と同様に、HTML エディタによって構成される。本実施例に係る命令付与手段は、入力される文書内に、抽出命令に代えて属性抽出命令を、挿入置換命令に代えて属性置換命令を、それぞれ埋め込む。各命令は、次のように記述される。すなわち、

## 【 0 1 6 9 】

## 【数 3 1】

・属性抽出命令

<!--#GET LABEL タグ名-->

※HTML で記述された文書部品

<!--#/GET-->

## 【 0 1 7 0 】

## 【数 3 2】

・属性置換命令

<!--#PUT LABEL タグ名 属性名-->

※HTML で記述された文書部品

<!--#/PUT-->

## 【 0 1 7 1 】

上記において、'GET' が属性抽出命令を、'PUT' が属性置換命令を、それぞれ表わすものとする。ここで、<!--#GET LABEL タグ名-->及び<!--#/GET-->をそれぞれ属性抽出命令開始タグ及び属性抽出命令終了タグとし、同様に、<!--#PUT LABEL タグ名 属性名-->及び<!--#/PUT-->をそれぞれ属性置換命令開始タグ及び属性

置換命令終了タグとする。

【0172】

属性抽出命令を付与したHTMLファイルの例を以下に示す。本明細書中では、該HTMLファイルを第3の原料文書とする。

【0173】

【数33】

```
<HTML>
<HEAD>
<TITLE> Sample 3 </TITLE>
</HEAD>
<BODY>
  <H2>リンク集</H2>
  <HR>
<!--#GET リンク A-->
  <A HREF="http://sample.com/" target="win0">リンク1</A>
<!--#GET-->
<HR>
</BODY>
</HTML>
```

【0174】

この第3の原料文書を標準的なHTMLブラウザで表示した様子を図22に示しておく。

【0175】

また、属性置換命令を付与したHTMLファイルの例を以下に示す。本明細書では、該HTMLファイルを第3の雛型文書とする。

【0176】

【数34】

```
<HTML>
```

```

<HEAD>
<TITLE> Template3 </TITLE>
</HEAD>
<BODY>
  <H1>資料集</H1>
  <HR>
  <IMG SRC="images/chart.gif" alt="チャート" WIDTH="600" HEIGHT="200" BORDER="0" USEMAP="#mapdata0">

  <MAP NAME="mapdata0">
<!--#PUT リンク AREA=HREF-->
    <AREA SHAPE="rect" ALT="リンク集" coords=0, 20, 200, 85 HREF="">
<!--#/PUT-->
  </MAP>
  <HR>
</BODY>
</HTML>

```

【0177】

命令付与手段及び入力手段の構成及び動作特性は上記した第1の実施例と略同一なので、ここでは説明を省略する。

【0178】

解析手段についても第1の実施例と同様である。但し、本実施例に係る解析手段は、注釈タグについては以下の規則に従うか否かを解析する。すなわち、

【0179】

【数35】

"#GET<空白><LABEL文字列><空白><タグ名文字列>" → 属性抽出命令開始タグ

” # /GET” → 属性抽出命令終了タグ

” # PUT<空白><LABEL文字列><空白><タグ名文字列>=<属性名文字列>” → 属性置換命令開始タグ

” # /PUT” → 属性置換命令終了タグ

【0180】

図11には第3の原料文書の解析木を、また、図12には第3の雛型文書の解析木をそれぞれ示している。

【0181】

図11において、属性抽出命令[0 1 2]が付与された文書部品は[0 1 3]のノードである。同様に、図12において属性置換命令[0 1 3 0]が付与された文書部品は[0 1 3 1]のノードである。これら文書部品中において、属性抽出命令又は属性置換命令で指定されたタグ名のタグをそれぞれ属性抽出命令、属性置換命令の対象タグとする。本実施例では、属性抽出命令の対象タグは<A>タグであり、また、属性置換命令の対象タグは<AREA>タグである。文書部品中に複数の対象タグがあってもよい。対象タグの検索は、上記した第1の実施例の場合と同様に、ツリーを行きがけ順に巡回して行い、引数で指定されたタグ名のタグを集めることができる。

【0182】

命令分離手段もまた、第1の実施例のそれと略同一である。但し、第1の実施例とは相違して、オペコードは属性抽出命令であるGETと、属性挿入命令であるPUTとなる。また、属性抽出命令にはタグ名、属性挿入命令にはタグ名と属性名がそれぞれ該命令の引数として付与されているため、それらをオペランドとして出力する。

【0183】

命令分離手段によって、第3の原料文書及び第3の雛型文書の各々から分離した命令列を以下に示す。但し、文書解析木のIDをそれぞれ5及び6とする。

【0184】

【数36】

第3の原料文書から分離した命令列

オペコード	ラベル	タグ名	文書部品 I D
GET	リンク	A	<5, [0 1 3] >

【0185】

【数37】

第3の雛型文書から分離した命令列

オペコード	ラベル	タグ名	属性名	文書部品 I D
PUT	リンク	AREA	HREF	<6, [0 1 3 1] >

【0186】

文書処理記述合成手段は、複数の原料文書または雛型文書から分離した命令列を入力し、命令列を併合し、順序の変更をし、変換と合成を行なって文書処理記述を出力する。以下で説明する点を除いては、文書処理記述合成手段は第1の実施例と同様である。

【0187】

#### ■処理C： 命令の合成

原料文書のGETと雛型文書のPUTのすべての組み合わせのうち、ラベル文字列が一致するものでペアを構成し、これらのペアから新しい命令を合成する。この動作手順について以下に説明する。

【0188】

本実施例では、例えば”GET LABEL タグ名1 文書部品1”と”PUT LABEL タグ名2 属性名 文書部品2”というペアから”SUBST タグ名1 文書部品1 タグ名2 文書部品2 属性名”を合成する。SUBSTは、文書部品1に含まれるタグ名1のノードの属性フィールドから、属性名で指定した属性値を選択して、文書部品2に含まれるタグ名2のノードの属性フィールドに設定する命令である。この動作によって、雛型文書に含まれるタグの属性値を別のものに置き換えることが可能になる。属性値の置き換えは、タグ名が異なるペアであっても同じ名前の属性フィールドがあれば行われる。

【0189】

【数38】

Step 41: 原料文書のすべての命令列を走査し、ラベル文字列ごとに命令の配列を作成する。この配列をGET配列と呼ぶ。GET配列は、ラベル文字列をキーとする連想配列に格納しておく。

Step 42: 雛型文書のすべての命令列を走査し、ラベル文字列ごとに命令の配列を作成する。この配列はPUT配列と呼ぶ。PUT配列は、ラベル文字列を属性として保持する。

Step 43: すべてのPUT配列についてStep 4以降を実行する。

Step 4: PUT配列の属性であるラベル文字列と一致する文字列をオペランドに持つGET配列を取り出す。

Step 45: PUT配列のすべての要素に対して、0番目から順にSTEP 6以降を実行する

Step 46: GET配列のm番目の要素として格納されたGET命令オペランド中のタグ名であるタグ名1と、文書部品IDである文書部品1を取り出す。PUT配列のm番目の要素として格納されたPUT命令オペランド中のタグ名であるタグ名2と属性名と文書部品IDである文書部品2を取り出す。

Step 47: 文書処理記述に次の命令を追加する。この例では、以下に示すように、オペコードはSUBSTであり、また、オペランドはタグ名1、文書部品1、タグ名2、文書部品2、及び属性名である。

SUBST タグ名1 文書部品1 タグ名2 文書部品2 属性名

【0190】

上記の処理によって、第3の原料文書と第3の雛型文書とから合成される文書処理記述の例を以下に示しておく。

【0191】

【数39】

SUBST A 5, [0 1 3] AREA 6, [0 1 3 1] HREF

【0192】

この命令は、文書5（図11を参照のこと）のノード[0 1 3]以下のAタグのHREF属性の値を文書6（図12を参照のこと）のノード[0 1 3



1] 以下の AREA タグの HREF 属性に設定することを指示するものである。

【0193】

図 1 1 及び図 1 2 に示した例では、原料文書中の文書部品に GET 命令が付与されている場合を対象としたが、本実施例に係る構造化文書処理システムは、GET 命令が雛型文書中の文書部品に付与されている場合であっても同様の処理を実現することができる。その場合、前述した命令の合成処理の Step 4 2 において、GET 命令の配列と PUT 命令の配列を別個に作成して、GET 命令の配列を Step 4 1 で作成した配列に追加すればよい。

【0194】

また、上述では置換する属性名がただ 1 つの場合について説明したが、属性置換開始命令中に属性名を” , ” 記号で区切って複数記述することによって、複数の属性値を一度に置き換えるようにしてもよい。

【0195】

次いで、インタプリタと属性抽出手段、及び属性置換手段の動作について説明する。

【0196】

インタプリタの基本動作は第 1 の実施例のそれと略同一であるので、相違する点を中心に説明する。インタプリタは、入力された文書処理記述のオペコードが SUBST であれば、オペランドで指定された文書部品 1 の文書部品 ID とタグ名 1 を属性抽出手段へ入力する。また、文書部品 2 の文書部品 ID とタグ名 2、及び属性名を属性置換手段へ入力する。

【0197】

属性抽出手段は、インタプリタから入力された文書部品 ID から、指定された文書解析木のノードをアドレスに従って取り出す。そして、そのノード以下から、入力されたタグ名に一致する対象タグ 1 のノードを行きがけ順に探索して、最初に見つかったノードを返す。

【0198】

属性置換手段は、インタプリタから入力された文書部品 ID から、指定され

た文書解析木のノードをアドレスに従って取り出す。そして、そのノード以下のすべてのノードの中から、入力されたタグ名に一致する対象タグ2のノードを行きがけ順に探索する。そして、見つかった対象タグ2のノードの属性フィールドの中で、インタープリタから入力された属性名と一致するものの属性値を、対象タグ1の属性名のもの属性値で置き換える。対象タグ2が複数ある場合は、この置き換える処理を繰り返す。

## 【0199】

また、前記Step46とStep47に代えて、以下のStep46aとStep47aとしてもよい。

## 【0200】

Step46a: GET配列のm番目の要素として格納されたGET命令オペランド中のタグ名であるタグ名1と、文書部品IDである文書部品1を取り出す。文書部品1のノードから、解析木を行きがけ順に巡回して、タグ名1の対象タグ1を検索する。PUT配列のm番目の要素として格納されたPUT命令オペランド中のタグ名であるタグ名2と属性名と文書部品IDである文書部品2を取り出す。文書部品2のノードから、解析木を行きがけ順に巡回して、タグ名2の対象タグ2を検索する。

Step47a: 文書処理記述に次の命令を追加する。この例では、オペコードはSUBSTで、オペランドは、対象タグ1（の文書部品ID）、対象タグ2（の文書部品ID）、属性名である。

SUBST    対象タグ1    対象タグ2    属性名

## 【0201】

上記の処理によって、第3の原料文書と、第3の雛型文書とから合成される別の文書処理記述例を示す。

## 【0202】

## 【数40】

SUBST    5,    [0   1   3]   6,    [0   1   3   1]   HREF

## 【0203】

この命令は、文書5（図11を参照のこと）のノード[0   1   2]の(Aタ

グの) HREF属性値を文書6 (図12を参照のこと) のノード [0 1 3 1] の (AREAタグの) HREF属性に設定するものである。

【0204】

インタプリタは、入力された文書処理記述のオペコードSUBSTのオペランドで指定された対象タグ1の文書部品IDと属性名を属性抽出手段へ入力する。また対象タグ2の文書部品IDと属性名を属性置換手段へ入力する。

【0205】

属性抽出手段と属性置換手段は、入力された文書部品IDからアドレスに従って対象タグ1又は対象タグ2を取り出した後、対象タグ1の指定された属性名の属性値を取り出し、または対象タグ2の属性値と置換する。

【0206】

第3の原料文書と第3の雛型文書とから合成した結果のHTMLファイルを以下に示す。

【0207】

【数41】

```
<HTML>
<HEAD>
<TITLE>  Template 3  </TITLE>
</HEAD>
<BODY>
  <H1>資料集</H1>
  <HR>
  <IMG SRC=" images/chart.gif" alt=" チャート" WIDTH=" 600" HEIGHT=" 200" BORDER=" 0" USEMAP=" #mapdata0" >

  <MAP NAME=" mapdata0" >
<!--#PUT リンク AREA=HREF-->
  <AREA SHAPE=" rect" ALT=" リンク集" coor
```

```

d s = 0 , 2 0 , 2 0 0 , 8 5   H R E F = " h t t p : / / s a m p l e . c
o m / " >
< ! -- # / P U T -- >
< / M A P >
< H R >
< / B O D Y >
< / H T M L >

```

【 0 2 0 8 】

また、第 3 の原料文書と第 3 の雛型文書とから合成した結果の HTML ファイルを標準的な HTML ブラウザで表示した様子を図 2 3 に示しておく。

# d

第 3 の雛型文書の例では、表示される画像（チャート：i m a g e s / c h a r t . g i f）の指定された座標位置（c o o r d s = 0 , 2 0 , 2 0 0 , 8 5）をクリックすることにより、リンク先のページをたどっていくことができる。本実施例の文書処理によって、第 3 の雛型文書では指定されていなかったリンク先が第 3 の原料文書で指定されていたもの（h t t p : / / s a m p l e . c o m）に置換されている。

【 0 2 0 9 】

次いで、本発明の第 3 の実施例について説明する。本実施例では、抽出命令に代えてパス名付き抽出命令が、反復複写命令に代えてパターン式付き反復複写命令が、挿入置換命令に代えてパターン式付き挿入置換命令が、それぞれ付与された構造化文書に対して文書処理を実行するものである。以下、第 1 の実施例との相違点を中心に本実施例に関して説明する。

【 0 2 1 0 】

命令付与手段は、第 1 の実施例と同様に、HTML エディタによって構成される。本実施例に係る命令付与手段は、入力される文書内に、抽出命令に代えてパス名付き抽出命令を、反復複写命令に代えてパターン式付き反復複写命令を、挿入置換命令に代えてパターン式付き挿入置換命令を、それぞれ埋め込む。各命令は、次のように記述される。すなわち、

【 0 2 1 1 】

【数 4 2】

・パス名付き抽出命令

< ! -- # O U T    パス名 -- >

※HTMLで記述された文書部品

< ! -- # / O U T -- >

【 0 2 1 2 】

【数 4 3】

・パターン式付き反復複写命令

< ! -- # R P    パターン式 -- >

※HTMLで記述された文書部品

< ! -- # / R P -- >

【 0 2 1 3 】

【数 4 4】

・パターン式付き挿入置換命令

< ! -- # I N    パターン式 -- >

※HTMLで記述された文書部品

< ! -- # / I N -- >

【 0 2 1 4 】

入力手段、解析手段、及び命令分離分離手段の各々は第 1 の実施例と略同一である。但し、命令分離手段については、ラベル文字列の代りにパス名またはパターン式をオペランドとして抽出する。これは、上記第 1 の実施例において説明した正規表現による文字列マッチを用いて容易に実現可能であるので、ここでは詳しく説明しない。

【 0 2 1 5 】

本実施例では、第 1 の原料文書（図 3 を参照のこと）のラベル文字列の「グループ」及び「サマリ」に代えて、パス名として「月報／グループ A」及び「月報／サマリ A」にした構造化文書を第 4 の原料文書（原料文書 4）として用いる。また、第 2 の原料文書（図 4 を参照のこと）のラベル文字列「グループ」及び「

サマリ」に代えて、パス名「月報／グループB」及び「月報／サマリB」にした構造化文書を第5の原料文書（原料文書5）とする。また、第2の原料文書のラベル文字列「グループ」及び「サマリ」に代えて、パス名「月報／グループE」及び「月報／サマリE」にした構造化文書を第6の原料文書（原料文書6）とする。

## 【0216】

本実施例では、パターン式は正規表現とする。

## 【0217】

また、第1の雛型文書のラベル文字列の「グループ」及び「サマリ」に代えて、パターン式として「. \* / グループ [A-D]」「. \* / サマリ [A-D]」にした構造化文書を第4の雛型文書（雛型文書4）とする。ここで言う正規表現「. \* / グループ [A-D]」の意味は、任意の文字列（. \*）の後に「 / グループ」が続き、次に、「A」か「B」か「C」か「D」のいずれかが続く文字列とマッチするパターンである。

## 【0218】

命令分離手段によってそれぞれの文書から分離した命令列を以下に示す。

## 【0219】

## 【数45】

第4の原料文書から分離した命令列

オペコード	ラベル	文書部品ID
OUT	月報／グループA	<7, [0 1 0 1]>
OUT	月報／サマリA	<7, [0 1 4]>

## 【0220】

## 【数46】

第5の原料文書から分離した命令列

オペコード	パス名	文書部品ID
OUT	月報／グループB	<8, [0 1 0 2]>
OUT	月報／サマリB	<8, [0 1 4]>

## 【0221】

## 【数 4 7】

第 6 の原料文書から分離した命令列

オペコード	パス名	文書部品 I D
OUT	月報／グループ E	< 9, [ 0 1 0 2 ] >
OUT	月報／サマリ E	< 9, [ 0 1 4 ] >

【 0 2 2 2 】

## 【数 4 8】

第 4 の雛型文書から分離した命令列

オペコード	パターン式	文書部品 I D
RP	. * / グループ [ A - D ]	< 1 0, [ 0 1 1 2 ] >
IN	. * / グループ [ A - D ]	< 1 0, [ 0 1 1 2 0 1 ] >
IN	. * / サマリ [ A - D ]	< 1 0, [ 0 1 1 2 1 1 ] >

【 0 2 2 3 】

次いで、本実施例の文書処理記述合成手段について説明する。但し、本実施例では、ラベル文字列の一致に代えて、パス名とパターン式のパターン・マッチを行なう以外は、第 1 の実施例における文書処理記述合成手段と略同一の処理を行なう。ここでは、原料文書 4 と原料文書 5 と雛型文書 4 のそれぞれから分離した命令列から文書処理記述を構成する処理を例にとってその動作特性について説明することにする。

## 【 0 2 2 4 】

## ■処理 A : OUT の計数

第 1 の実施例の処理 A で行なった OUT の計数は行なわない。

## 【 0 2 2 5 】

## ■処理 B : RP による各命令の複写

雛型文書 1 の命令列において、命令列中に RP が含まれていた場合、すべての RP のオペランドの文書部品中に含まれる IN と OUT と RP を、原料文書の OU

T命令のうち、パス名とパターン式のパターン・マッチングを行なって、マッチが成功した個数に従って複写する。この複写は各1つの雛型文書について、次のStep 51～Step 56を実行することで行われる。正規表現のパターン・マッチングは、例えば、UNIX上のgrepコマンド（周知）などを用いることで実現される。

【0226】

【数49】

Step 51：RPリストを作成する。

Step 52：RPリストから、対象とするRP（反復複写命令）を順に変更しつつ各RPについてStep 53以降を実行する。

Step 53：対象Rpのオペランドのパターン式を取り出し、命令列を先頭から順に走査してOUTのオペランドのパス名を取り出し、パターン・マッチングを行なって、成功したOUT命令の個数を反復複写回数tとする。tが2以上ならStep 54以降を実行する。

Step 54以降は、第1の実施例におけるStep 24以降と略同一なので、ここでは説明を省略する。

【0227】

パターン式「.\*/グループ[A-D]」は、「月報/グループA」と「月報/グループB」とのパターン・マッチは成功するが、「月報/グループE」とのパターンマッチは失敗する。したがって、上述の例では複写回数t=2となる。

【0228】

【数50】

第4の雛型文書から分離した命令列に対して上記の処理Bを施した結果得られる命令列

オペコード	パス名/パターン式	文書部品ID
RP	2	<10, [0 1 1 2]>
IN	.* /グループ[A-D]	<10, [0 1 1
2 0 1]>		
IN	.* /サマリ[A-D]	<10, [0 1 1



2 1 1] >

IN . \* / グループ [A-D] < 1 0, [0

1 1 3 0 1] >

IN . \* / サマリ [A-D] < 1 0, [0 1 1

3 1 1] >

【0 2 2 9】

#### ■処理C： 命令の合成

原料文書中のOUT命令と雛型文書中のIN命令とのすべての組み合わせのうち、パス名とパターン式のパターン・マッチングが成功するものどうしでペアを構成し、これらのペアから新しい命令を合成する。以下、この命令の合成動作について説明する。

【0 2 3 0】

本実施例では、例えば”OUT パス名 文書部品1”と”IN パターン式 文書部品2”というペアから”FILL 文書部品1 文書部品2”を合成する。FILLは、文書部品2を文書部品2と置換する命令を表す。この動作によって、原料文書に出現する文書部品の出現順にかかわらず、雛型文書の中でユーザが所望する任意の位置に、原料文書の文書部品を置換又は挿入することが可能になる。

【0 2 3 1】

【数51】

Step 61： 原料文書のすべての命令列を走査して、OUT命令を配列に格納する。この配列はOUT配列と呼ぶ。OUT配列は、パス名を属性として保持する。

Step 62： 雛型文書のすべての命令列を走査し、IN命令を配列に格納する。これをIN配列と呼ぶ。IN配列は、パターン式を属性として保持する。

Step 63： IN配列のすべてのIN命令について、Step 64以降を順次実行する。

Step 64： IN配列の属性であるパターン式を取り出し、OUT配列の要素を先頭から順に走査して、要素のパス名とパターン・マッチングを行ない、成功

した要素からOUT命令を取り出す。成功したOUT配列の要素は取り除く。

Step 65: OUT配列から取り出したOUT命令のオペランドの文書部品のIDである文書部品1を取り出す。また、IN命令のオペランドの文書部品のIDである文書部品2を取り出す。

Step 66: 文書処理記述に次の命令を追加する。この例では、命令のオペコードはFILLで、オペランドは文書部品1及び文書部品2である

FILL 文書部品1 文書部品2

【0232】

上記の処理手順によって文書処理記述が合成される。合成された文書処理記述の例を以下に示す。

【0233】

【数52】

※第1の原料文書と第2の原料文書と第1の雛型文書から合成された文書処理記述例1

RP	2	2, [0 1 1 2]
FILL	0, [0 1 0 1]	2, [0 1 1 2 0 1]
FILL	1, [0 1 0 2]	2, [0 1 1 3 0 1]
FILL	0, [0 1 4]	2, [0 1 1 2 1 1]
FILL	1, [0 1 4]	2, [0 1 1 3 1 1]

【0234】

上述した例では、パターン・マッチに失敗したOUT命令のオペランドで指定された文書部品は、雛型文書に挿入又は置換が行われないことに留意されたい。

【0235】

本実施例に係る構造化文書処理システムによって、原料文書4と原料文書5と原料文書6と雛型文書4から合成されるHTMLファイルを以下に示す。

【0236】

【数53】

<HTML>

<HEAD>

```

<TITLE> Template1 </TITLE>
</HEAD>
<BODY BGCOLOR=" #404040" >
    <H1>進捗サマリ</H1>
    <TABLE BORDER=" 1" >
        <TR><TH>チーム</TH><TH>概況</TH><
/TH>
<!--#RP . */グループ[A-D] -->
    <TR>
        <TH>
<!--#IN . */グループ[A-D] -->Aチーム<!--#/IN-
->
        </TH>
        <TD>
<!--#IN . */サマリ[A-D] --><UL>
            <LI> 項目1の進捗60% 計画どおり </L
I>
            <LI> 項目2の進捗30% 遅れ: 3日 </
LI>
            <LI> 項目3の進捗70% 予定外の問題発生、
計画見直しが必要 </LI>
        </UL><!--#/IN-->
        </TD>
    </TR>
    <TR>
        <TH>
<!--#IN . */グループ[A-D] -->Bチーム<!--#/IN-
->
        </TH>

```

```

<TD>
<!--#IN . */サマリ[A-D]--><TABLE BORDER=
" 0" CELLSPACING=" 0" CELLPADDING=" 1" >
    <TR><TH>項目</TH><TH>進捗</T
H><TH ALIGN=" LEFT" ">評価</TH></TR>
    <TR><TH>1</TH><TD>20%</T
D><TD>やや計画おくれ(1日)</TD></TR>
    <TR><TH>2</TH><TD>40%</T
D><TD>遅れなし</TD></TR>
    <TR><TH>3</TH><TD>60%</T
D><TD>進み: 3日</TD></TR>
</TABLE><!--#/IN-->
</TD>
</TR>
<!--#/RP-->
</TABLE>
</BODY>
</HTML>

```

【0237】

図24には、上記のHTMLファイルを標準的なHTMLブラウザ上で表示した様子を示している。

【0238】

本発明の第3の実施例は、第1の実施例と同様の作用効果を奏することができる。但し、ラベル文字列に代えてパターン式を用いたことにより、より柔軟で複雑な文書処理記述を合成することが可能となる点には充分理解されたい。

【0239】

次いで、本発明の第4の実施例について説明する。本実施例は、文書処理記述合成手段と挿入置換手段に代えて、以下に説明する文書処理記述合成手段と挿入置換手段を有するという点で、上記の第3の実施例とは顕著に相違する。

## 【 0 2 4 0 】

本実施例における命令文書処理記述合成手段が実行する処理Cについて以下に説明する。

## 【 0 2 4 1 】

## ■処理C： 命令の合成

原料文書中のOUT命令と雛型文書中のIN命令とのすべての組み合わせのうち、パス名とパターン式のパターン・マッチングが成功するものどうしでペアを構成し、これらペアから新しい命令を合成する。以下、この命令の合成動作について説明する。

## 【 0 2 4 2 】

本実施例では、例えば” OUT パス名 文書部品1 ”と” IN パターン式 文書部品2 ”というペアから” FILL 文書部品1 文書部品2 パス名 ”を合成する。FILLは、文書部品2を文書部品2と置換する命令を表す。FILLは、文書部品1を文書部品2と置換する命令を表す。加えて、挿入置換命令開始タグと挿入置換命令終了タグを、抽出命令開始タグと抽出命令終了タグにそれぞれ置換する命令を表す。この動作によって原料文書に出現する文書部品の出現順にかかわらず、雛型文書中でユーザが所望する任意の位置に、原料文書の文書部品を置換又は挿入することが可能になる。加えて、処理をした結果合成された文書を入力とし、本実施例に係る構造化文書処理システムによって再び処理する際には、元の原料文書に含まれる文書部品を識別することが可能になる。

## 【 0 2 4 3 】

## 【数54】

Step 71：原料文書のすべての命令列を走査し、OUT命令を配列に格納する。この配列をOUT配列と呼ぶ。OUT配列は、パス名を属性として保持する。

Step 72：雛型文書のすべての命令列を走査し、IN命令を配列に格納する。これをIN配列と呼ぶ。IN配列は、パターン式を属性として保持する。

Step 73：IN配列中のすべてのIN命令について、Step 74以降を順次実行する。

Step 74 : IN配列の属性であるパターン式を取り出し、OUT配列の要素を先頭から順に走査して、要素のパス名とパターン・マッチングを行ない、成功した要素からOUT命令を取り出す。成功したOUT配列の要素は取り除く

Step 75 : OUT配列から取り出したOUT命令のオペランドの文書部品のIDである文書部品1を取り出す。加えて、OUT命令のパス名を取り出すとともに、IN命令のオペランドの文書部品のINである文書部品2を取り出す。

Step 76 : 文書処理記述に次の命令を追加する。この例では、オペコードはFILLで、オペランドは文書部品1と文書部品2とパス名である

FILL      文書部品1      文書部品2      パス名  
【0244】

パス名は、原料文書から分離した抽出命令（IN命令）のオペランドから取り出したパス名文字列に加えて、所定の名前の環境変数やグローバル変数に代入されている文字列、あるいは、本実施例に係る構造化文書処理システムを起動した際に引数として引き渡される文字列を、適当な区切り文字で区切って付与してもよい。付与する前記文字列を履歴文字列と呼ぶ。上述した例では、区切り文字として” / ”を用い、履歴文字列としてホスト名を与えている。但し、ホスト名は” 総務部 ”とする。この処理によって合成された文書処理記述の例を示す。

【0245】

【数55】

※第4の原料文書と第5の原料文書と第4の雛型文書から合成された文書処理記述例

RP	2	2, [0 1 1 2]	
FILL	0, [0 1 0 1]	2, [0 1 1 2 0 1]	
総務部／月報／グループA			
FILL	1, [0 1 0 2]	2, [0 1 1 3 0 1]	
総務部／月報／グループB			
FILL	0, [0 1 4]	2, [0 1 1 2 1 1]	総
務部／月報／サマリA			
FILL	1, [0 1 4]	2, [0 1 1 3 1 1]	総

務部／月報／サマリ B

【0 2 4 6】

次いで、インタプリタと挿入置換手段について説明する。

【0 2 4 7】

インタプリタは、文書処理記述の F I L L 命令を解釈して、挿入置換手段に、文書部品 1 と文書部品 2 とパス名を入力する。これ以外の点については実施例 1 のインタプリタと略同一である。

【0 2 4 8】

挿入置換手段は、文書部品 2 で指定される I D の文書部品にもともと付与されている挿入置換命令である挿入置換命令開始タグと挿入置換命令終了タグを、それぞれ抽出命令開始タグと抽出命令終了タグに置き換える。また、インタプリタから入力されたパス名を抽出命令開始タグに付与する。この置き換え処理は、H T M L ファイル中のそれぞれの注釈タグの注釈文字列を置き換えることによって容易に実現できる。この点以外は、上述した第 1 の実施例における挿入置換手段と略同一である。

【0 2 4 9】

本実施例に係る構造化文書処理システムによって、原料文書 4 と原料文書 5 と雛型文書 4 から合成される H T M L ファイルを以下に示す。

【0 2 5 0】

【数 5 6】

<HTML>

<HEAD>

<TITLE> T e m p l a t e 1 </TITLE>

</HEAD>

<BODY BGCOLOR=" # 4 0 4 0 4 0 " >

<H1>進捗サマリ</H1>

<TABLE BORDER=" 1 " >

<TR><TH>チーム</TH><TH>概況</TH><

/TR>

<!--#RP . \*/グループ[A-D]-->

<TR>

<TH>

<!--#OUT 総務部/月報/グループA-->Aチーム<!--#/OUT  
T-->

</TH>

<TD>

<!--#OUT 総務部/月報/サマリA--><UL>

<LI> 項目1の進捗60% 計画どおり </L  
I>

<LI> 項目2の進捗30% 遅れ: 3日 </  
LI>

<LI> 項目3の進捗70% 予定外の問題発生、  
計画見直しが必要 </LI>

</UL><!--#/OUT-->

</TD>

</TR>

<TR>

<TH>

<!--#OUT 総務部/月報/グループB-->Bチーム<!--#/OU  
T-->

</TH>

<TD>

<!--#OUT 総務部/月報/サマリB--><TABLE BORDER  
=" 0" CELLSPACING=" 0" CELLPADDING=" 1"  
>

<TR><TH>項目</TH><TH>進捗</T  
H><TH ALIGN=" LEFT" ">評価</TH></TR>

<TR><TH>1</TH><TD>20%</T



```

D><TD>やや計画おくれ（1日）</TD></TR>
      <TR><TH>2</TH><TD>40%</T
D><TD>遅れなし</TD></TR>
      <TR><TH>3</TH><TD>60%</T
D><TD>進み： 3日</TD></TR>
      </TABLE><!--#OUT-->
      </TD>
    </TR>
  <!--#RP-->
</TABLE>
</BODY>
</HTML>

```

【0251】

図25には、上記のHTMLファイルを標準的なHTMLブラウザ上で表示した様子を示している。

【0252】

本発明の第4の実施例は、第1の実施例と同様の作用効果を奏することができるとともに、さらに、原料文書の抽出命令とパス名を合成結果の構造化文書の文書部品に継承して付与することができる。

【0253】

本実施例に係る構造化文書処理システムを複数段接続してパイプラインを構成することによって、最初の構造化文書処理システムで出力された原料文書に含まれている文書部品は、2段目以降の構造化文書処理ではパス名を調べることで識別することができる。この機能によって、より柔軟で複雑な構造化文書処理を実現することが可能となる。

【0254】

次いで、本発明の第5の実施例について説明する。

【0255】

上述した第1の実施例では、構造化文書処理システムは、HTTPリクエスト

を入力して、所定の構造化文書の処理を行ない、合成した構造化文書をHTTPレスポンスとして出力するHTTPサーバとして構成される。これに対し、本実施例では、HTTPリクエストに処理起動記述が埋め込まれている。ここで言う処理起動記述は、原料文書のURLと雛型文書のURLの双方を含むものとする。

## 【0256】

本実施例に係る構造化文書処理システムは、このような処理起動記述を解析して、原料文書のURLと雛型文書のURLとを取り出す処理起動記述解析手段をさらに具備している。そして、処理起動記述解析手段によって解析した結果として取り出されたURLに基づいて、入力手段により原料文書や雛型文書を入力して処理することができる。図13には、本実施例に係る構造化文書処理システムの構成を模式的に示している。以下、第1の実施例と相違する点を中心に該システムの構成及び動作特性について説明する。

## 【0257】

処理起動記述は、HTTPリクエストに、構造化文書処理装置の原料文書のURLと、雛型文書のURLを埋め込むことで構成される。本実施例では、特に、HTTPリクエストに含まれているURL文字列中に、所定の文法形式で、原料文書のURLと雛型文書のURLが埋め込まれる。

## 【0258】

HTTPリクエストのURL中に別のURLを埋め込んだものを「ヴァーチャルURL」と呼ぶ。すなわち、本実施例に係る処理起動記述はヴァーチャルURLとして構成される。なお、処理起動記述の文法と、該文法に従って埋め込む方法と取り出す方法に関しては、例えば、本出願人に既に譲渡されている特許第2746218号公報（特開平8-292910号）「資源管理装置および資源管理方法」に詳述されているので、必要であれば参照されたい。

## 【0259】

処理起動記述解析手段は、HTTPリクエストを所定の文法形式に従って解析して、原料文書のURLと雛型文書のURLの各々を取り出す。構造化文書処理システムにおける処理起動記述解析手段以外の機能ブロックの構成及び動作特性

は、第 1 の実施例と略同一である。

【0 2 6 0】

構造化文書処理サーバは、HTTP リクエストの URL 中に原料文書の URL と雛型文書の URL を含んだ形式の処理起動記述を受け取って、この処理起動記述に指定された原料文書と雛型文書を入力して文書処理記述を合成して、さらに該文書処理記述を解釈して構造化文書処理を行なうことができる。

【0 2 6 1】

本実施例において利用される処理起動記述の例を以下に示す。

【0 2 6 2】

【数 5 7】

原料文書 1 の URL : `http://host1/Sample1.html`

原料文書 2 の URL : `http://host2/Sample2.html`

雛型文書 1 の URL : `http://host3/Templat1.html`

構造化文書処理サーバの URL : `http://server/HTMLTools?method=fill`

処理起動記述 (HTTP リクエストの URL) :

`http://server/HTMLTools?method=fill&template=http%3A%2F%2Fhost3%2FTemplat1.html&file1=http%3A%2F%2Fhost1%2FSample1.html&file2=http%3A%2F%2Fhost2%2FSample2.html`

【0 2 6 3】

上記に示した処理起動記述の例では、雛型文書の URL は、文字列 "template=" 以降で、次の "&" の前までに埋め込まれている。元々の URL に含まれていた ":" と "/" は、それぞれ "%3A" と "%2F" にエンコードされ変換されている。また、原料文書 1 と原料文書 2 に関する URL についても同様に、それぞれ文字列 "file1=" 以降から次の "&" の前まで、並びに、文字列 "file2=" 以降に埋め込まれている。

## 【0264】

特許第2746218号公報に示すようなURLの表記方式によれば、2段階に構造化文書処理サーバを起動するヴァーチャルURLを階層的に構成することができる。例えば、原料文書や雛型文書は既に他の構造化文書処理サーバで処理した結果であってもよい。但し、この場合の原料文書や雛型文書のURLは、第2の構造化文書処理サーバのURLに基づくヴァーチャルURLである。

## 【0265】

## 【数58】

原料文書3のURL： 前記HTTPリクエストのURL（処理結果のヴァーチャルURL）

雛型文書2のURL： URL： `http://host3/Template2.html`

第2の構造化文書処理サーバのURL： `http://server2/HTMLTools?method=fill`

前記文書処理記述を原料文書とする第2の処理起動記述（第2のHTTPリクエストのURL）：

`http://server2/HTMLTools?method=fill  
&template=http%3A%2F%2Fhost3%2FTemplate2.html&file1=http%3A%2F%2Fserver%2FHTMLTools%3Fmethod%3Dfill%3Ftemplate%3Dhttp%253A%252F%252Fhost3%252FTemplate1.html%26file1%3Dhttp%253A%252F%252Fhost1%252FSample1.html%26file2%3Dhttp%253A%252F%252Fhost2%252FSample2.html`

## 【0266】

図14には、上述したような第2の処理起動記述によって起動される処理の流れを示している。

## 【0267】

第2の構造化文書処理サーバ（server2）には、雛型文書2（Template2.html）を用いて原料文書3を処理する旨のHTTPリクエストが入力される。このHTTPリクエストはヴァーチャルURLを用いて記述される。

【0268】

雛型文書2（Template2.html）は、WWWサーバ（host3）で提供されており、通常のURL表記”http://host3/Template2.html”によって指定される。

【0269】

他方、原料文書3は、第1の構造化文書処理サーバ（server1）によって処理された結果である。より具体的には、該処理結果とは、WWWサーバ（host1）の資源である原料文書1（Sample1.html）と、WWWサーバ（host2）の資源である原料文書2（Sample2.html）とを、WWWサーバ（host3）の資源である雛型文書1（Template1.html）を用いて処理したものである。

【0270】

したがって、第1の構造化文書処理サーバは、原料文書1、原料文書2、及び雛型文書1を各WWWサーバから入力して、その処理結果である原料文書3を第2の構造化文書処理サーバに渡す。

【0271】

次いで、第2の構造化文書処理サーバは、さらにWWWサーバ（host3）から原料文書2を入力して、原料文書3を処理して、この処理結果である構造化文書をHTTPレスポンスとして要求元に返す。

【0272】

図13に示したような構造化文書処理システムによれば、分散的に文書処理命令を管理し処理することができる。したがって、文書の改定や処理システムの変更に対して、より柔軟な文書処理を提供することができる。文書を処理するサーバをモジュールとして取り扱うことができるので、原料文書や雛形文書と文書処理モジュールを文書処理記述上で組み合わせることで、エンド・ユーザが容易に

カスタマイズすることができる柔軟な文書処理システムを構築することができる。

【0273】

次いで、本発明の第6の実施例について説明する。

【0274】

本実施例は、上記第5の実施例に係る第1の構造化文書処理サーバ及び第2の構造化文書処理サーバが、単一のデータベースを共有する構成を備えているものである。

【0275】

データベースには、識別子としてURL又は処理起動記述をキーとして、このURL又は処理起動記述に対応する構造化文書の解析木が格納されている。したがって、URL又は処理起動記述を指定することで、所望の構文解析木を取り出すことができる。このようなデータベースは、URL又は処理起動記述から適当なハッシュ値を計算し、解析木を実現するデータへのポインタを格納したハッシュ法による表の探索で実現できる（ハッシュ法については、例えば「岩波書店アルゴリズムとデータ構造」2.7節 p123-137に詳述されている）。

【0276】

図15には、本実施例に係る第1の合成サーバの機能ブロック図を示している。本実施例においては、第2の（すなわち後続の）合成サーバは、第1の合成サーバと略同一の構成及び動作特性を有するものと理解されたい。

【0277】

第1の合成サーバは、HTTPリクエストを入力して、所定の構造化文書の処理を行ない、合成した構造化文書をHTTPレスポンスとして出力するHTTPサーバとして実現されている。本実施例は、以下の2点を除き、第1の実施例と略同一である。

【0278】

（1）抽出手段、反復複写手段、並びに置換挿入手段は、解析手段から構文解析木を入力する代わりに、切り替え手段を経由して構文解析木を入力する。

（2）切り替え手段は、システムの状態を調べて、解析手段から構文解析木を入

力するか、又は、保持手段から構文解析木を入力するかを切り替える。

【0279】

システムの状態は、例えば、処理中の処理起動記述をキーとして、保持手段の中に該当する構文解析木があるかどうかを調べることで判定できる。

【0280】

保持手段に該当する構文解析木が格納されていれば、改めて構造文書を解析する必要はないので、保持手段から構文解析木を入力するように切り替える。この場合、構文解析木は、図15中の記号（い）で示された一点破線の矢印方向に従って処理される。

【0281】

他方、保持手段に該当する構文解析木が格納されていない場合には、切り替え手段は、解析手段から新たに構文解析木を入力するように切り替える。この場合、図15中の記号（ろ）で示された一点破線の矢印方向に従って、入出力が行われる。

【0282】

本実施例に係る構造化文書処理システムの構成によれば、第1の合成サーバで処理した結果は、構文解析木のまま第2（後続）の合成サーバに入力することができる。したがって、出力手段による構造化文書のHTMLへの変換と、解析手段によるHTMLの解析という2つの処理を省略することができる。

【0283】

また、第1の合成サーバにおいて一旦計算した結果は保持手段に格納されるので、第2の合成サーバが第1の合成サーバに処理起動記述を入力して第1の合成サーバが改めて処理を行う代わりに、既に計算されている前記処理起動記述に対応する構文解析木を利用することで、第1の合成サーバでの計算や合成処理を省略することができる。

【0284】

したがって、従来に比べ処理性能の優れた効率的な分散文書処理を実現することができる。

【0285】

次いで、本発明の第 7 の実施例について説明する。

【0286】

図 16 には、本発明の第 7 の実施例に係る構造化文書処理システムの構成を模式的に示している。本実施例は、上述した第 2 の実施例の構成に対して、第 5 の実施例に係る処理起動記述解析手段をさらに追加したものである。

【0287】

第 2 の実施例では、属性抽出命令及び属性置換命令がそれぞれ指定していた文書部品の間で属性置換を行っていた。これに対し、本実施例では、属性抽出命令が文書部品ではなく手続き名を指定する。属性置換命令は、指定した文書部品の属性を、属性抽出命令で指定された手続きで変換したものに置き換える。処理起動記述解析手段については第 5 の実施例のそれと略同一であるので、以下では第 2 の実施例との相違点を中心に説明することにする。

【0288】

本実施例では、属性抽出命令を以下のような形式で記述する。

【0289】

【数 59】

・属性抽出命令

<!--#GET LABEL 手続き名-->

<!--#/GET-->

【0290】

第 2 の実施例の場合とは相違し、属性抽出命令には置換する対象のタグ名を指定する必要がなく、開始タグと終了タグの間に文書部品を挟む必要もない。属性置換命令は、第 2 の実施例と同一である。

【0291】

属性抽出命令を付与した HTML ファイルを第 4 の原料文書とし、以下に示す。

【0292】

【数 60】

<HTML>



```
<HEAD>
<TITLE> Sample 3 </TITLE>
</HEAD>
<BODY>
  <H2>リンク集</H2>
  <HR>
  <!--#GET リンク Proc1-->
  <!--# /GET-->
  <HR>
</BODY>
</HTML>
```

【0293】

解析手段は、注釈タグが以下の規則に従うかを解析する。

【0294】

【数61】

"#GET<空白><LABEL文字列><空白><手続き名文字列>" →

属性抽出命令開始タグ

"# /GET" → 属性抽出命令終了タグ

※属性置換命令については第2の実施例と同じ。

【0295】

属性抽出命令には対象タグがないので、文書部品から対象タグを探す処理も行わない。

【0296】

命令分離手段も第2の実施例と同様であるが、属性抽出命令にはタグ名ではなく手続き名が引数として付与されているため、これをオペランドとして出力する。命令分離手段によって第4の原料文書から分離した命令列を以下に示す。

【0297】

【数62】

オペコード	ラベル	手続き名
-------	-----	------

GET                    リンク                    P r o c 1

【0298】

文書処理記述合成手段は、処理Cすなわち命令の合成において、第2の実施例と相違する。文書処理記述合成手段は、属性抽出命令と属性置換命令とのペアから新しい命令を合成するが、例えば、"GET LABEL 手続き名"と"PUT LABEL タグ名 属性名 文書部品"のペアから、"SUBST 手続き名 タグ名 文書部品 属性名"を合成する。SUBSTは、文書部品に含まれるタグ名のノードの属性フィールドから属性名で指定した属性値を選択して、手続き名で指定した手続きで属性値を変換する命令である。

【0299】

処理手順は、Step 46以降が第2の実施例とは相違し、以下の通りとなる。

【0300】

【数63】

Step 46 b: GET配列のm番目の要素として格納されたGET命令オペランド中の手続き名を取り出す。PUT配列のm番目の要素として格納されたPUT命令オペランド中のタグ名と属性名、文書部品IDである文書部品を取り出す。

Step 47 b: 文書処理記述に次の命令を追加する。この例では、オペコードはSUBSTで、オペランドは、手続き名、タグ名、文書部品、属性名である。

SUBST 手続き名 タグ名 文書部品 属性名

【0301】

上記の処理によって、第4の原料文書と第3の雛型文書とから合成される文書処理記述例を以下に示す。

【0302】

【数64】

SUBST    P r o c 1    AREA    6, [0   1   3   1]    H R E  
F

【0303】

この命令は、文書 6 のノード [ 0 1 3 1 ] 以下の AREA タグの HREF 属性の値を Proc 1 で変換したものに置き換えることを指示するものである。

#### 【 0 3 0 4 】

インタプリタは、SUBST 命令を属性置換手段に入力する。本実施例の属性置換手段の動作は、第 2 の実施例のそれとは異なり、SUBST 命令を解釈して次のように動作する。すなわち、対象タグの HREF 属性の属性値を第 1 の引数として、手続き Proc 1 を呼び出して、Proc 1 の戻り値として得られた文字列を新たな HREF 属性として対象タグに設定する。但し、手続きの呼び出しは普通に知られる手続き呼び出し（サブルーチン・コール）であり、HREF 属性の取り出しや HREF 属性の設定は通常の文字列の分離又は結合である。

#### 【 0 3 0 5 】

属性値を変換する手続きとしては、例えば次のようなものが考えられる。属性置換命令を付与した文書部品が属性値フィールドに URL を設定できる種類のタグのノードを含んでいる場合には、その URL 文字列を第 5 の実施例で説明した処理起動記述（ヴァーチャル URL）に変換する。例えば、元の URL に設定されている雛型文書と文書を特定する文字列を解析し、それらを元に新たな処理起動記述を合成して返すような手続きである。このような URL 文字列は、手続きの引数として受け渡される。

#### 【 0 3 0 6 】

ここで、変換前の URL の記述例と変換後の処理起動記述の例を示す。但し、この例では、処理起動記述解析手段に入力された処理起動記述が含んでいる原料文書のうち 2 番目のものを、新たに合成する処理起動記述に含める原料文書とする。また、新たに合成する処理起動記述に含める雛型文書は、入力された処理起動記述が含んでいる雛型文書とは別のものにする。

#### 【 0 3 0 7 】

##### 【数 6 5】

元の処理起動記述（処理起動記述解析手段に入力されたもの）：

`http://server/HTMLTools?method=fill&`

template=http%3A%2F%2Fhost3%2FTemplate  
te1.html&file1=http%3A%2F%2Fhost1%2F  
Sample1.html&file2=http%3A%2F%2Fhost  
2%2FSample2.html

【0308】

【数66】

雛型文書のURL: http://host3/Template2.h  
tml

変換前のURL(): http://host3/Template2.h  
tml?file2=

【0309】

【数67】

変換後のURL(処理起動記述):

http://server/HTMLTools?method=fill&  
template=http%3A%2F%2Fhost3%2FTempla  
te2.html&file2=http%3A%2F%2Fhost2%2F  
Sample2.html

【0310】

また、手続きの別の例として、新たに処理起動記述を合成するのではなく、元  
の処理起動記述に含まれる原料文書のURLをそのまま返すようなものも考えら  
れる。

【0311】

変換前のURLの記述例と変換後の処理起動記述の例を以下に示す。但し、処  
理起動記述解析手段に入力された処理起動記述が含んでいる原料文書のうちの最  
初のものを原料文書とする。

【0312】

【数68】

元の処理起動記述(処理起動記述解析手段に入力されたもの):

http://server/HTMLTools?method=fill&

```
template=http%3A%2F%2Fhost3%2FTemplate1.html&file1=http%3A%2F%2Fhost1%2FSample1.html&file2=http%3A%2F%2Fhost2%2FSample2.html
```

【0313】

【数69】

変換前のURL ( ) : dummy.html#file1

【0314】

【数70】

変換後のURL (処理起動記述) :

```
http://host1/Sample1.html
```

【0315】

上記の例で示した変換前のURLは、ここで定義した独自の記述方法である。変換前のURLは、第3の雛型文書の例ではAREAタグのHREF属性値として設定すればよい。

【0316】

また、手続きの引数としてOSのシステム・コールを用いて得られる日時やシステムの状態変数などを利用するようにしてもよい。

【0317】

次いで、本発明の第8の実施例について説明する。

【0318】

図17には、本発明の第7の実施例に係る構造化文書処理システムの構成を模式的に示している。本実施例は、上述した第6の実施例の構成に対して、処理起動記述合成手段と、エラー通知文書合成手段をさらに追加したものである。

【0319】

本実施例の命令分離手段は、第1の実施例に係る命令分離手段が有する動作に加え、命令列を抽出する際に検出される命令の文法的なエラーを検出して、エラー情報をエラー通知文書合成手段に入力するようになっている。ここで言うエラー情報は、エラーの種類やエラーの検出された文書中の位置(行数)など、通常

のエラー処理でよく扱われるものとする（命令分離手段におけるエラーの検出やエラー処理の実現方法については、例えば、岩波講座ソフトウェア科学5 「プログラミング言語処理系」（岩波書店）の P 8 9, P 1 2 3 - 1 2 5, 2 0 6 - 2 0 8 に詳解されている）。

#### 【0 3 2 0】

また、本実施例の出力手段は、第1の実施例に係る出力手段が持つ機能に加えて、処理起動記述合成手段によって合成された処理起動記述を、出力結果としてのHTML形式文書に挿入するように動作するようになっている。この動作手順について以下に説明する。

#### 【0 3 2 1】

エラー通知文書合成手段は、エラー情報を入力して、エラー通知を行なう文書を合成する。エラー情報は例えばエラー番号とし、エラー文書はエラー番号をファイル名とするファイルとする。エラー通知文書合成手段は、命令解析手段から入力されたエラー番号をファイル名とみなして、ファイルからエラー通知文書の1つを入力する。入力された文書は、解析手段を用いて構文解析木が作られる。構文解析木を作る動作は、第1の実施例の場合と同様である。構文解析木は後述するように保持手段に保持されるが、必ずしも構文解析木を作る必要はなく、文字列情報のまま保持手段に保持してもよい。例えば、エラー番号1は、抽出命令開始タグと抽出命令終了タグの対応がとれないエラーであるとし、エラー通知文書のファイル名はファイル1. h t m l とする。ファイルの例を示す。

#### 【0 3 2 2】

##### 【数 7 1】

<HTML>

<HEAD>

<TITLE> エラー1</TITLE>

<BODY>

原料ファイルの処理中に抽出命令開始タグと抽出命令終了タグの対応がとれないエラーが発生した。

</BODY>

&lt;/HTML&gt;

【0323】

処理起動記述合成手段は、エラー通知ページをアクセスするためのURLを合成する。例えば、この文書処理を行なった合成サーバが“server”であって、文書処理を行なったプロセスのプロセス番号が12345番であったとすると、以下に示すようにプロセス番号を含むURLを合成する。

【0324】

【数72】

```
http://server/HTMLTools?method=userlog&pid=12345
```

【0325】

次いで、本実施例の出力手段の動作について、第1の実施例との相違点を中心に説明する。例えば、上述したような処理起動記述が入力されると、文字列<META USERLOG=" http://server/HTMLTools?method=userlog&pid=12345">を合成して、出力されるHTML形式の文書に挿入する。このような挿入は、通常知られる単純な文字列合成や文字列挿入処理である。挿入された結果のHTML形式の文書の一部（例7-A）を以下に例示しておく。

【0326】

【数73】

&lt;HTML&gt;

&lt;HEAD&gt;

```
<META USERLOG=" http://server/HTMLTools?method=userlog&pid=12345">
```

&lt;/HEAD&gt;

...

【0327】

また、別の文字列を合成して挿入した例7-Bを以下に示す。

【0328】

【数74】

&lt;HTML&gt;

&lt;HEAD&gt;

```
<SCRIPT LANGUAGE="JavaScript"> <!--
window.open("http://server/HTMLTools
?method=userlog&pid=12345");
```

// --&gt;

&lt;/SCRIPT&gt;

&lt;/HEAD&gt;

【0329】

エラー通知文書合成手段で合成されたエラー通知文書は、処理起動記述合成手段で合成されたURLをキー情報として、保持手段に格納され保持される。この保持手段に文書を格納する動作は第6の実施例とは相違する。

【0330】

上述した第6の実施例では、WWWクライアントからのリクエストすなわちHTTPリクエストとして、第1の処理起動記述が与えられる。次いで、処理起動記述処理が行なわれ、処理結果として合成された文書が第1の処理起動記述をキーとして保持手段に格納される。これに対し、本実施例では、後のエラー通知文書を得るための第2の処理起動記述を、エラー通知文書のリクエストに先行して合成サーバ側で合成し、（実際のリクエストが入力されるより前に）保持手段に格納するように動作するようになっている。

【0331】

合成サーバserverにおいて文書処理を実行した際にエラーが発生した場合を例にとって以下に説明してみる。

【0332】

上述した例7-Aのタイプの場合には、利用者は、serverから出力されるHTTPレスポンスによって送信されるHTML形式の文書の内容からUSERLOG属性を有するMETAタグを検索して、エラー通知文書の処理起動記述に相当するURLを得る。利用者は、このURLを含む第2のHTTPリクエス



トを `server` に送信する。合成サーバは、処理起動記述解析手段により HTTP リクエストに含まれる処理起動記述を取り出し、これをキーとして保持手段から文書を検索し、エラー通知文書を得る。それ以降は、第 1 の実施例の場合と同様の動作手順を実行して、新たな HTTP レスポンスとしてエラー通知文書を出力する。

#### 【0333】

エラー通知文書を雛型文書として取り扱えば、適当な文書処理が行われて、出力される。この場合、エラーが発生した位置（行数）やエラーが発生した文書の ID などを挿入することが考えられる。このような文書処理に対する引数（入力パラメータ）は、CGI（Common Gateway Interface）として当業界において広く知られている HTTP サーバに対する URL の記述方法に従って文書処理記述中に含めることができる。

#### 【0334】

また、例 7-B に示すようなタイプの場合には、URL を含む第 2 の HTTP リクエストを自動的に送信するスクリプト記述（例えば JavaScript 言語）を `server` が出力する（第 1 の）HTTP レスポンスによって送信される HTML 文書中に挿入することで、人手の介入をすることなく自動的にエラー通知文書を得ることができる。例えば、WWW ブラウザ側では、第 1 の HTTP レスポンスを受信すると、HTML 形式の文書に含まれるスクリプトを実行する。そのスクリプトは、前記処理起動記述を `server` に送って得られるエラー通知文書を、所定のウィンドウを開いて表示するものとなっている。それ以外の動作は、例 7-A と略同一であると理解されたい。

#### 【0335】

要言すれば、本実施例の構成では文書処理の途中で発生したエラーを、文書処理の合成結果とは別にして利用者に通知することができる。したがって、高度で柔軟な文書処理アプリケーションを構成できる。さらに、エラーが発生した時点でエラー通知文書を生成して、文書処理の結果を一時的に格納する保持手段に（エラー通知文書のリクエストに先行して）格納し、後のエラー通知文書の要求に対して出力するように動作することができる。このため、エラーが発生したプロ

セスとエラーの種類やエラーに関する情報を対応付けて保持するような機構を用意する必要がなくなる。

【0336】

〔追補〕

以上、特定の実施例を参照しながら、本発明について詳解してきた。しかしながら、本発明の要旨を逸脱しない範囲で当業者が該実施例の修正や代用を成し得ることは自明である。すなわち、例示という形態で本発明を開示してきたのであり、限定的に解釈されるべきではない。本発明の要旨を判断するためには、冒頭に記載した特許請求の範囲の欄を参酌すべきである。

【0337】

【発明の効果】

以上詳記したように、本発明によれば、文書を章、節、段落、図表、あるいは表題や章題、概要などの複数の素片に分解し各素片をノードとするツリー構造やグラフ構造などのように構造的に表現して取り扱うことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することができる。

【0338】

また、本発明によれば、複数の構造化文書を基にして新たに文書を合成することができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することができる。

【0339】

また、本発明によれば、複数の構造化文書から特定の条件を満たす文書部分（「文書部品」）を取り出すとともに文書部品を別の文書中に挿入又は置換して合成を行なうことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することができる。

【0340】

また、本発明によれば、構造化文書から文書部品を抽出したり、各文書部品を雛型文書中に挿入又は置換したりする手続きを記述したスクリプトを用いることなしに構造化文書の合成処理を行うことができる、優れた構造化文書処理システム及び構造化文書処理方法を提供することができる。

【図面の簡単な説明】

【図 1】

本発明に係る構造化文書処理システムの基本概念を示した機能ブロック図である。

【図 2】

本発明の第 1 の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。

【図 3】

第 1 の原料文書を解析木として表現した図である。

【図 4】

第 2 の原料文書を解析木として表現した図である。

【図 5】

第 1 の雛型文書を解析木として表現した図である。

【図 6】

第 2 の雛型文書を解析木として表現した図である。

【図 7】

雛型文書 1 に対して、文書処理記述例 1 の R P 命令で反復複写した結果の文書解析木の例を示した図である。

【図 8】

雛型文書 2 に対して、文書処理記述例 2 の R P 命令で反復複写した結果の文書解析木の例を示した図である。

【図 9】

第 1 の原料文書と第 2 の原料文書と第 1 の雛型文書から合成した文書の解析木を示した図である。

【図 1 0】

本発明の第 2 の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。

【図 1 1】

第 3 の原料文書の解析木を示した図である。

【図 1 2】

第 3 の雛型文書の解析木を示した図である。

【図 1 3】

本発明の第 5 の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。

【図 1 4】

第 2 の処理起動記述によって起動される処理の流れを示した図である。

【図 1 5】

本発明の第 6 の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。

【図 1 6】

本発明の第 7 の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。

【図 1 7】

本発明の第 8 の実施例に係る構造化文書処理システムの構成を模式的に示した機能ブロック図である。

【図 1 8】

第 1 の原料文書を標準的な HTML ブラウザで表示した結果を示した図である。

【図 1 9】

第 2 の原料文書を標準的な HTML ブラウザで表示した結果を示した図である。

【図 2 0】

第 1 の原料文書と第 2 の原料文書と第 1 の雛型文書から合成して生成された HTML ファイルを標準的な HTML ブラウザで表示した結果を示した図である。

【図 2 1】

第 1 の原料文書と第 2 の原料文書と第 2 の雛型文書から合成して生成された HTML ファイルを標準的な HTML ブラウザで表示した結果を示した図（第 1 の実施例）である。

【図 2 2】

第 3 の原料文書を標準的な HTML ブラウザで表示した結果を示した図である。

【図 2 3】

第 3 の原料文書と第 3 の雛型文書から合成して生成された HTML ファイルを標準的な HTML ブラウザで表示した結果を示した図（第 2 の実施例）である。

【図 2 4】

原料文書 4 と原料文書 5 と原料文書 6 と雛型文書 4 から合成される HTML ファイルを標準的な HTML ブラウザで表示した結果を示した図（第 3 の実施例）である。

【図 2 5】

原料文書 4 と原料文書 5 と雛型文書 4 から合成される HTML ファイルを標準的な HTML ブラウザで表示した結果を示した図（第 4 の実施例）である。

【図 2 6】

従来技術に係る構造文書処理システムの構成例を示した図である。

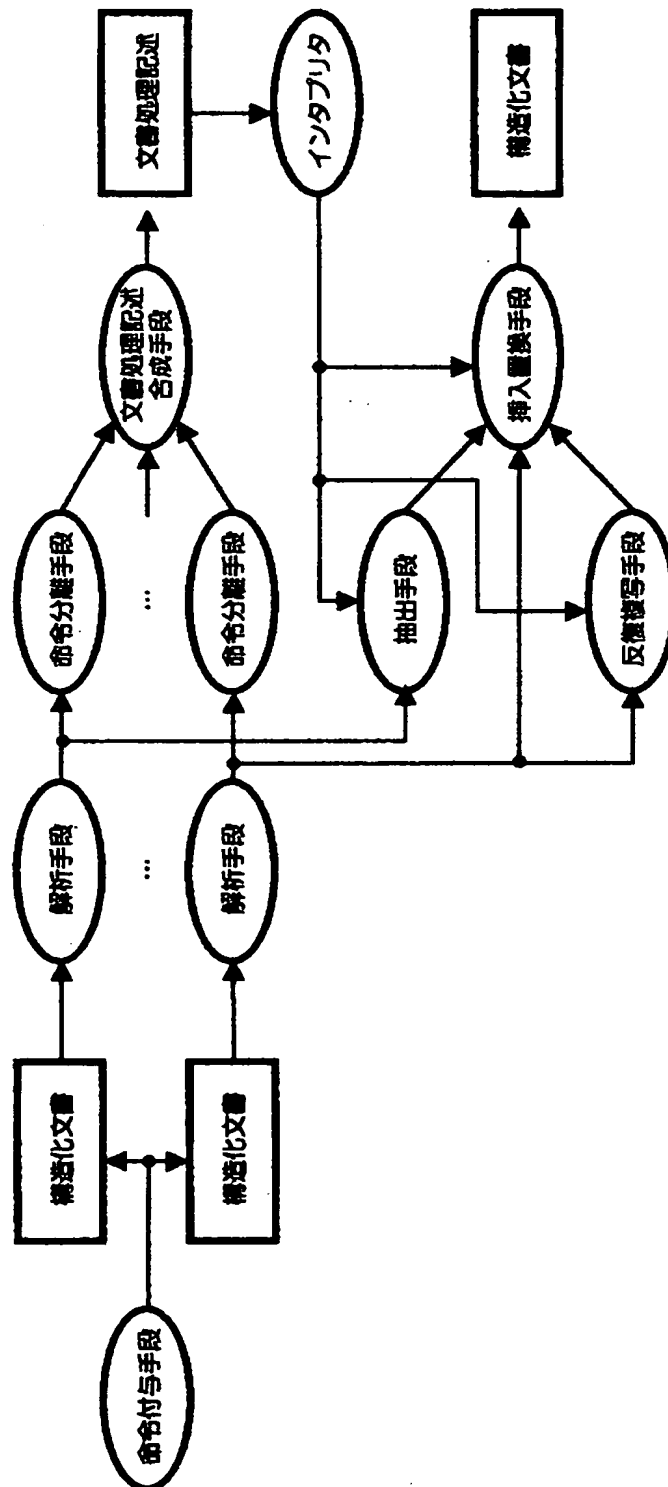
【図 2 7】

従来技術に係る構造文書処理システムの構成例を示した図である。

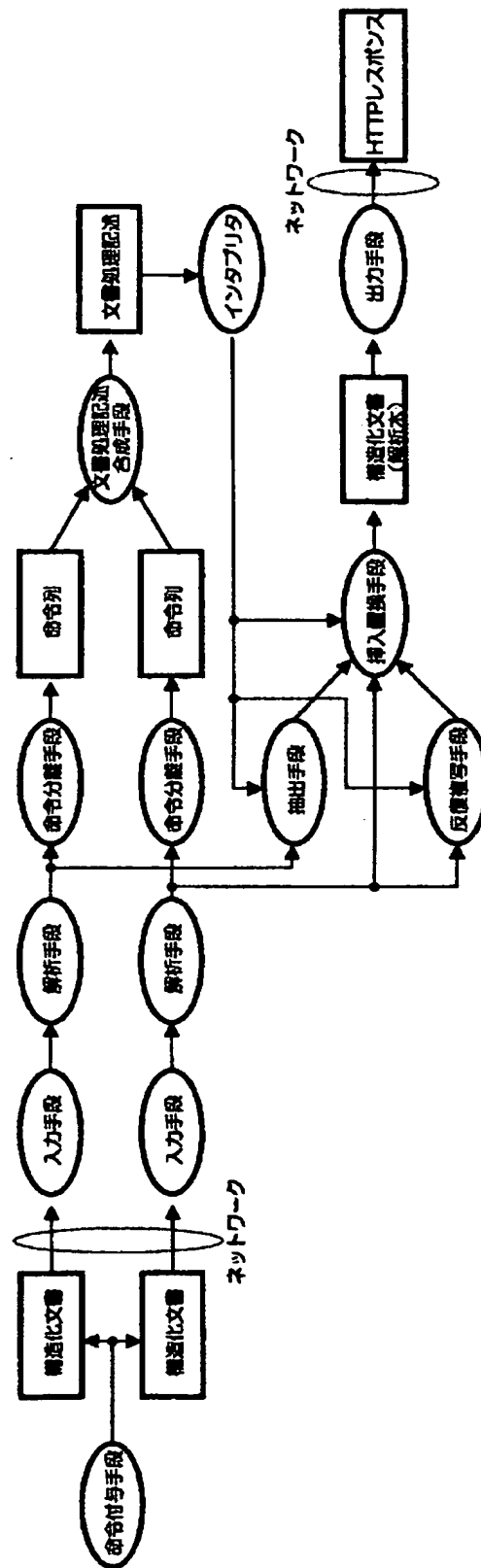
【書類名】

図面

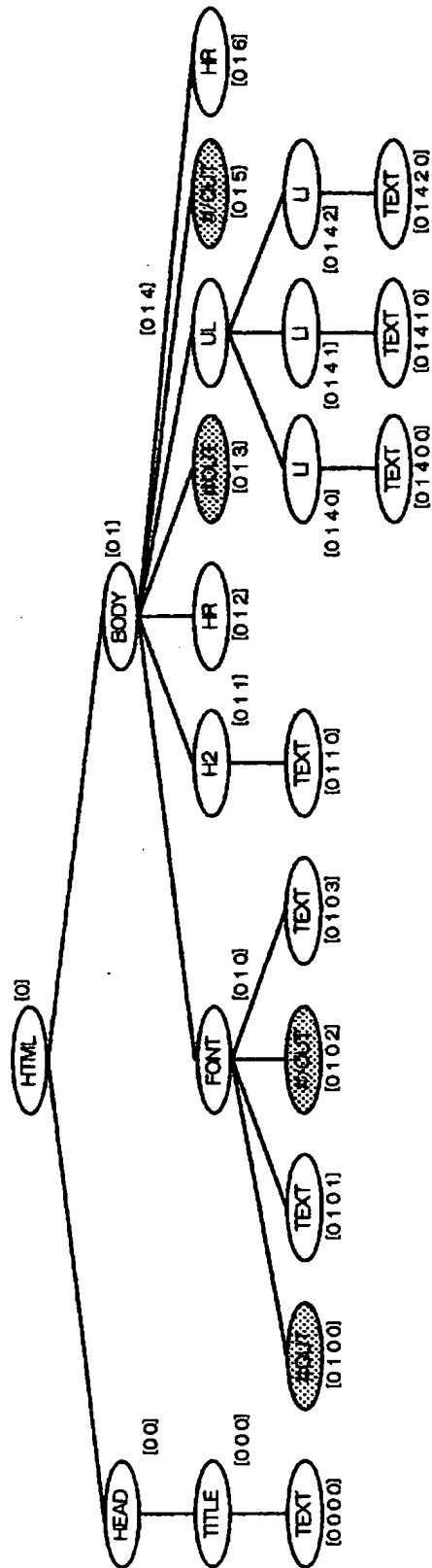
【図 1】



【図 2】

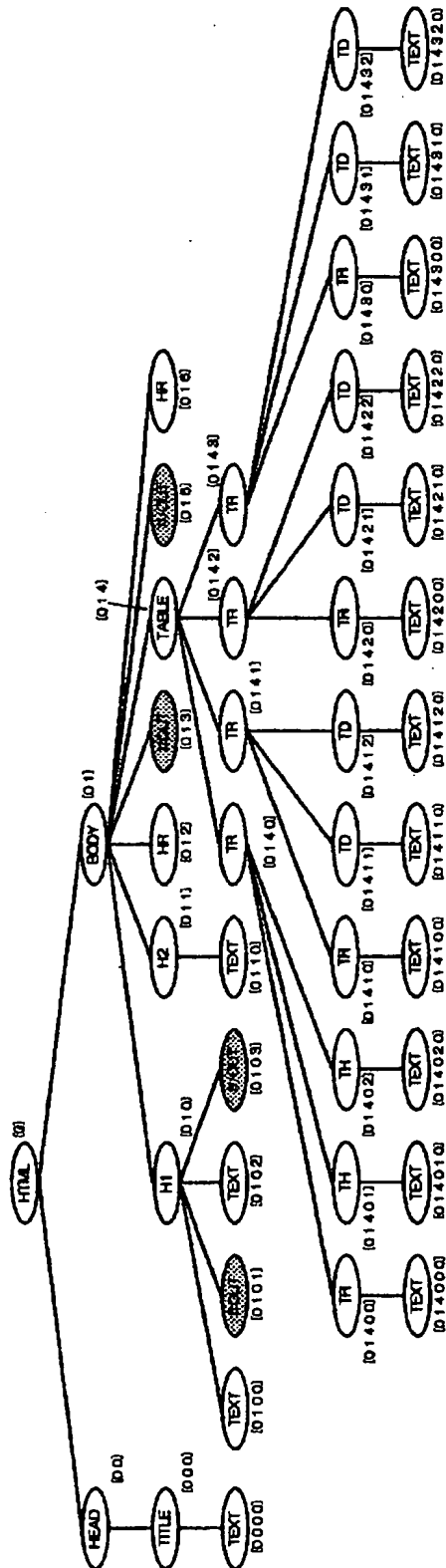


【図 3】

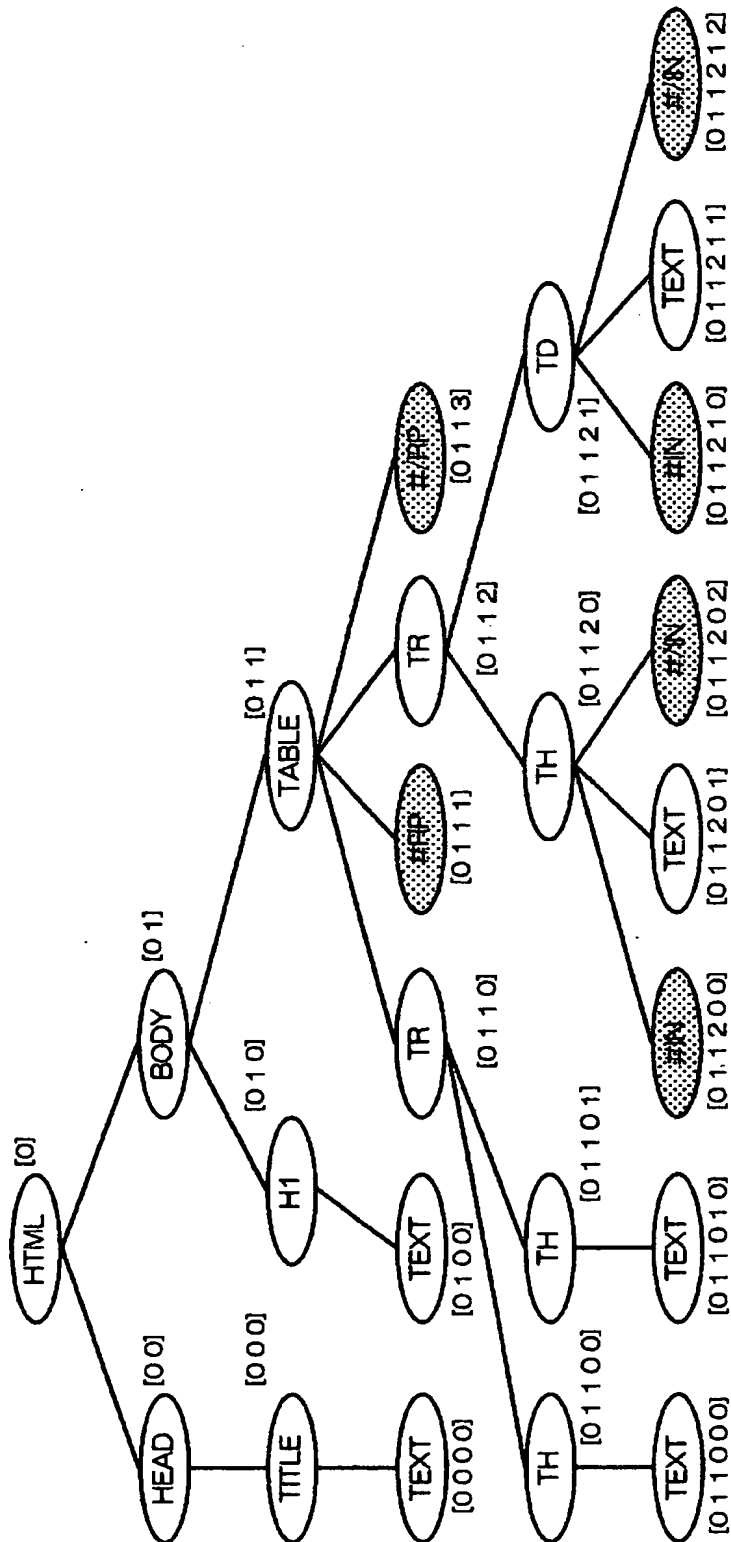




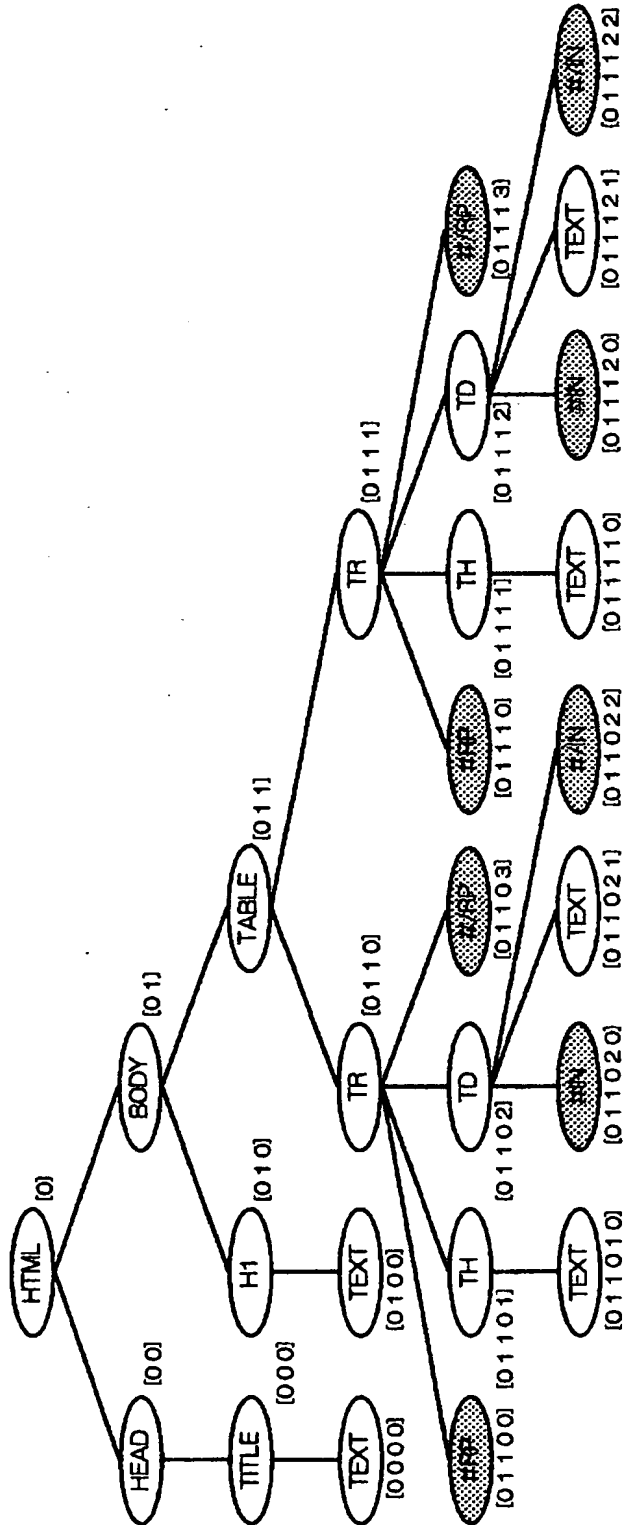
【図4】



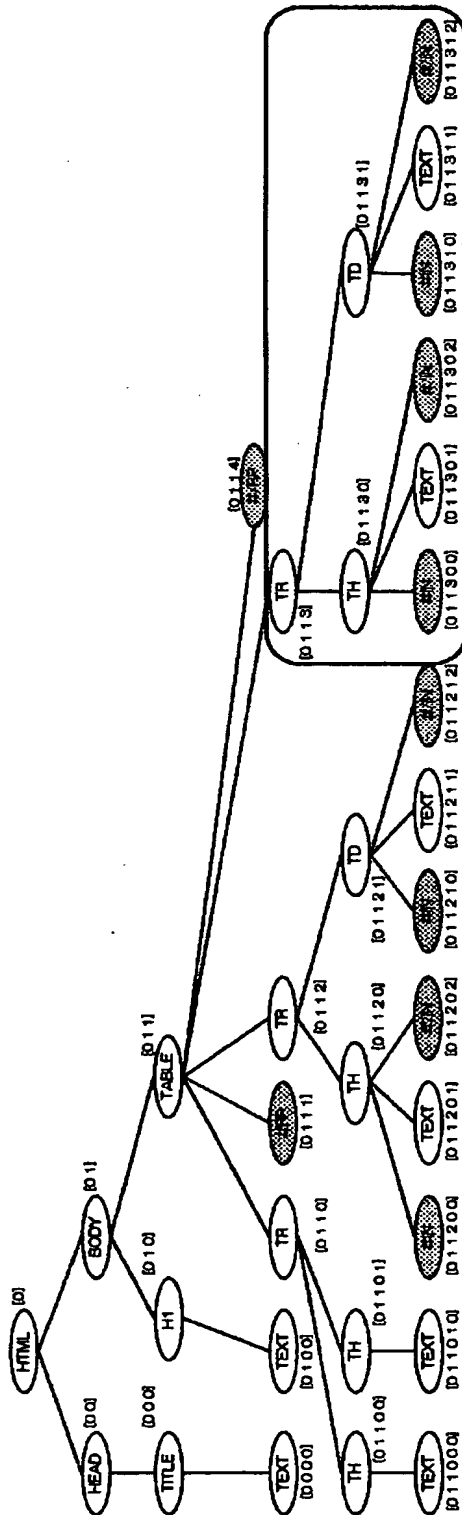
【图 5】



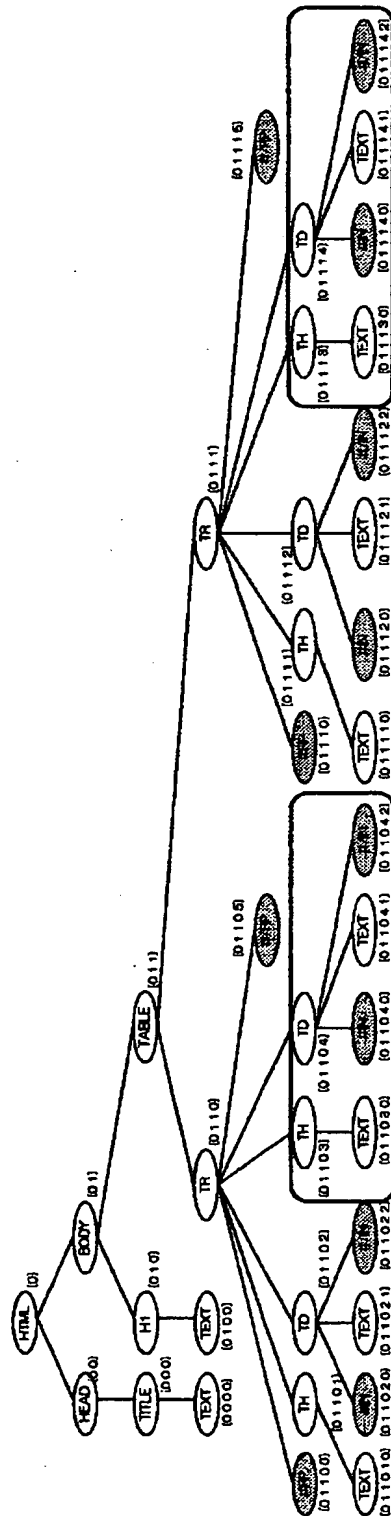
【図 6】



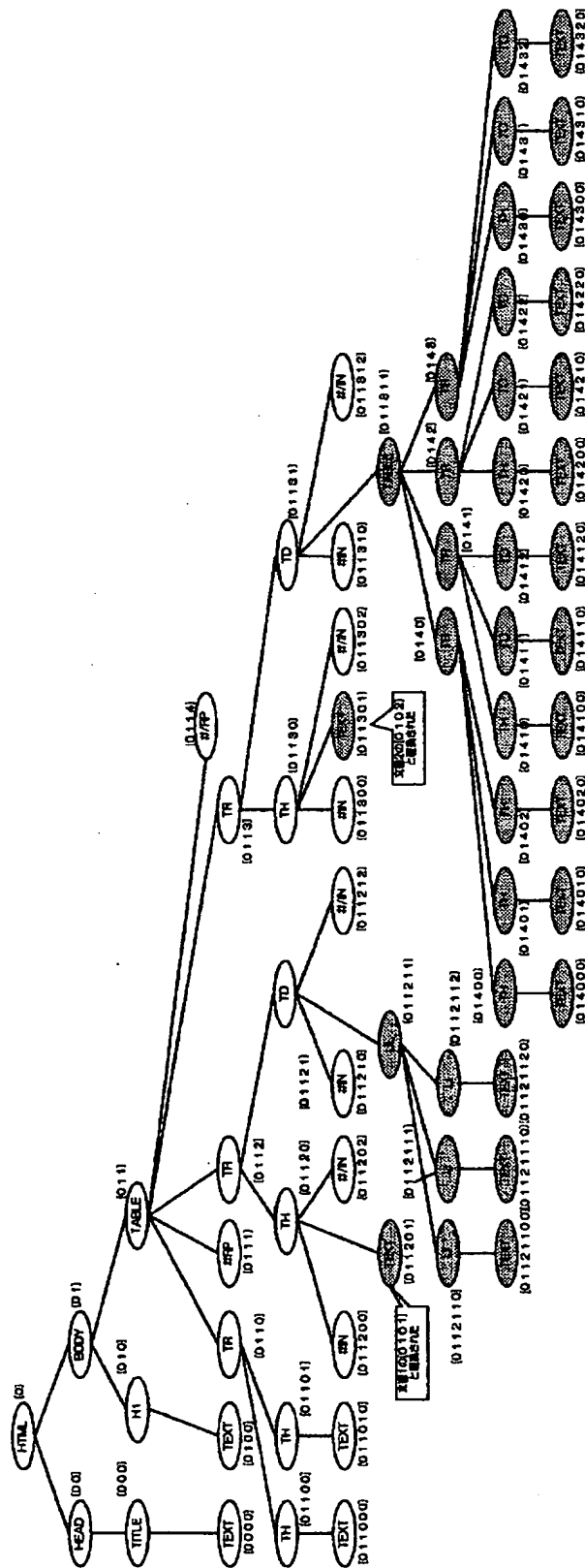
【图7】



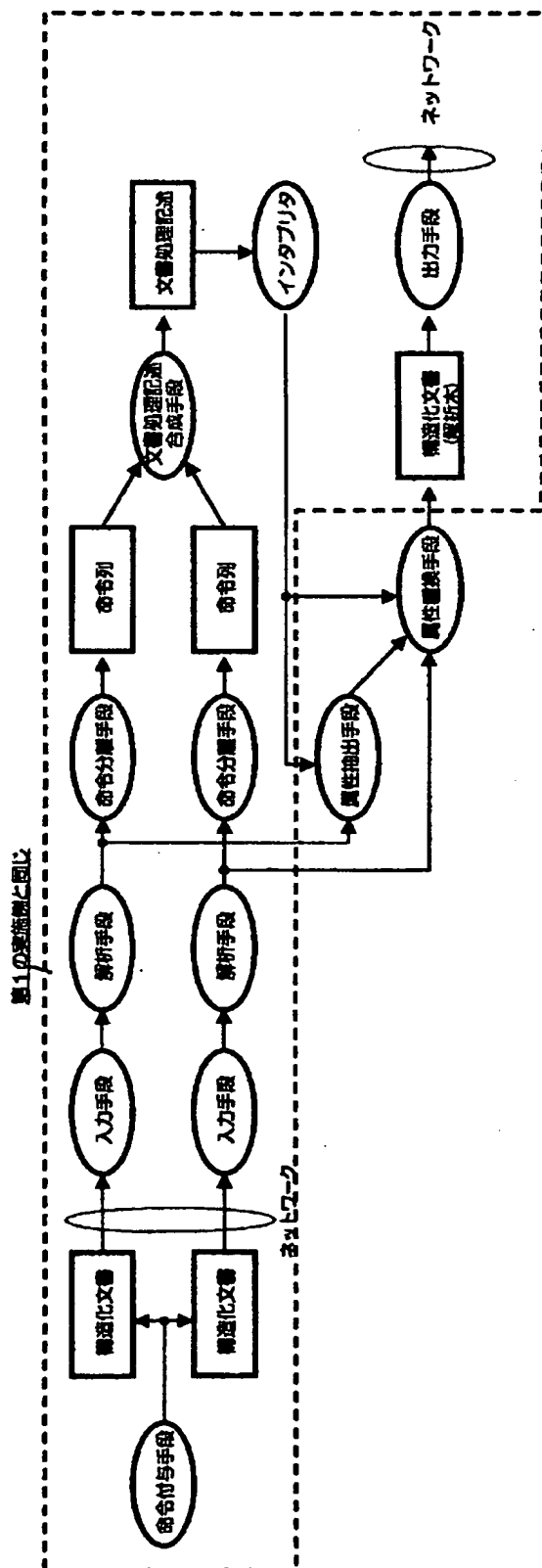
【図 8】



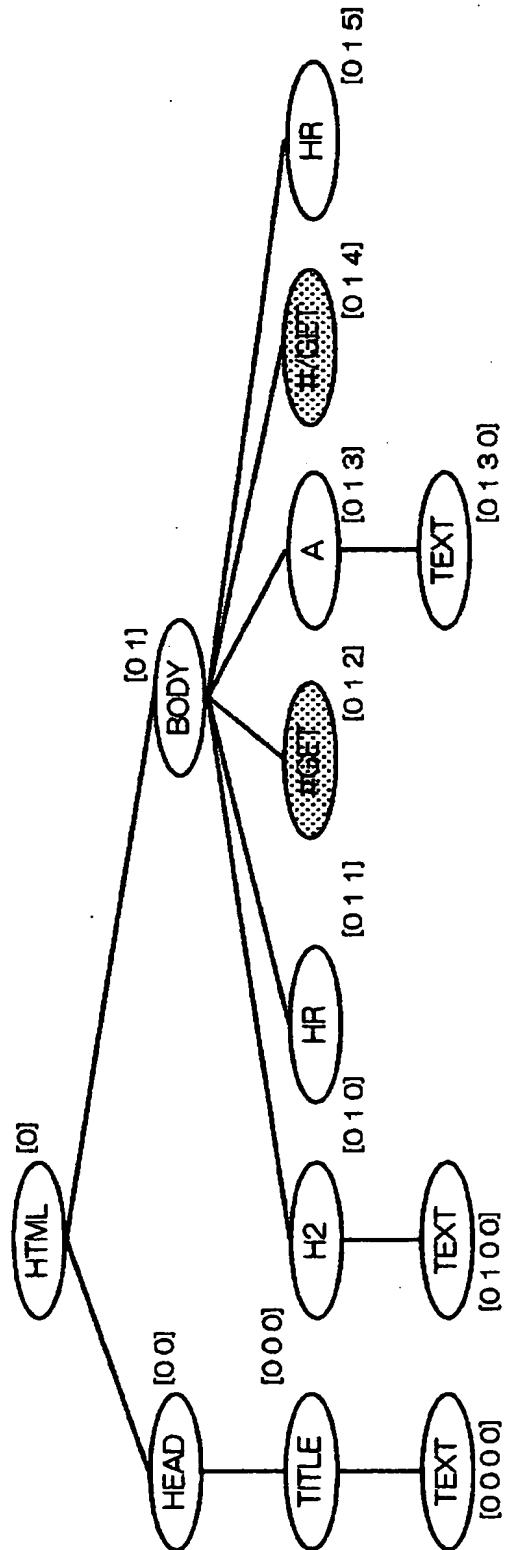
【図9】



【図 10】

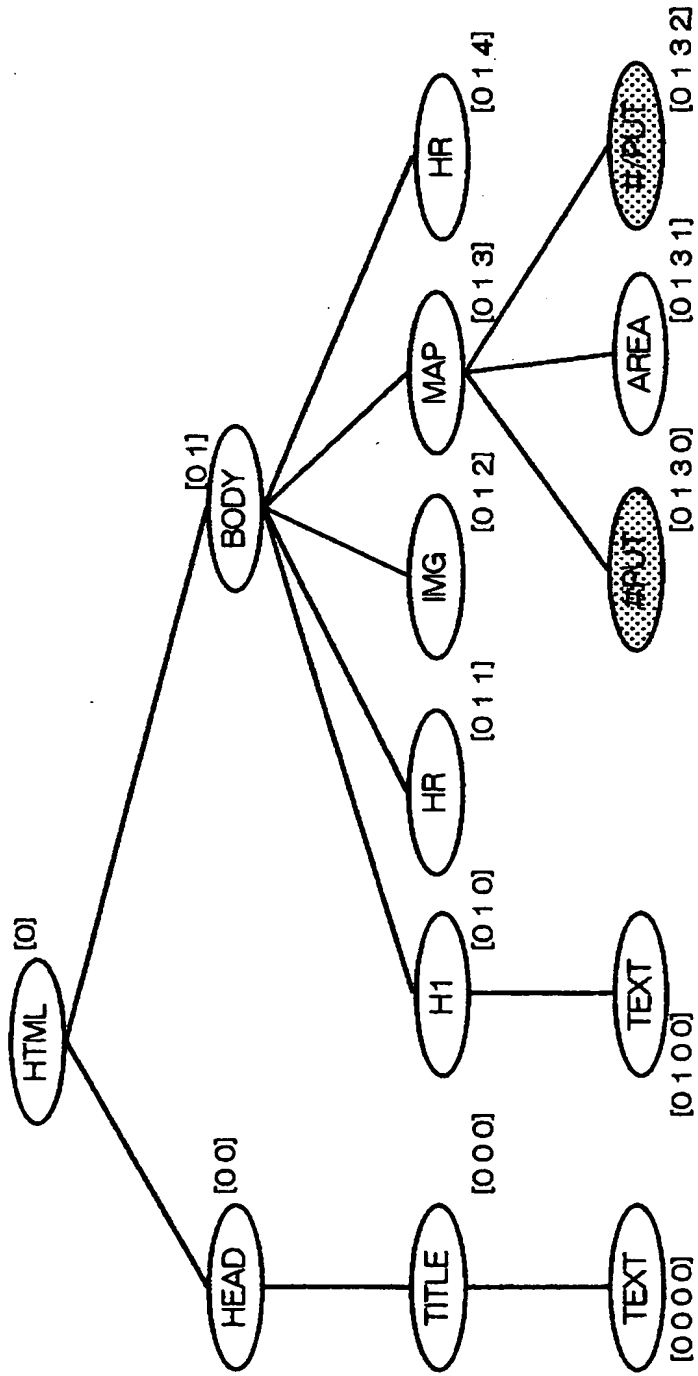


【図 1 1】

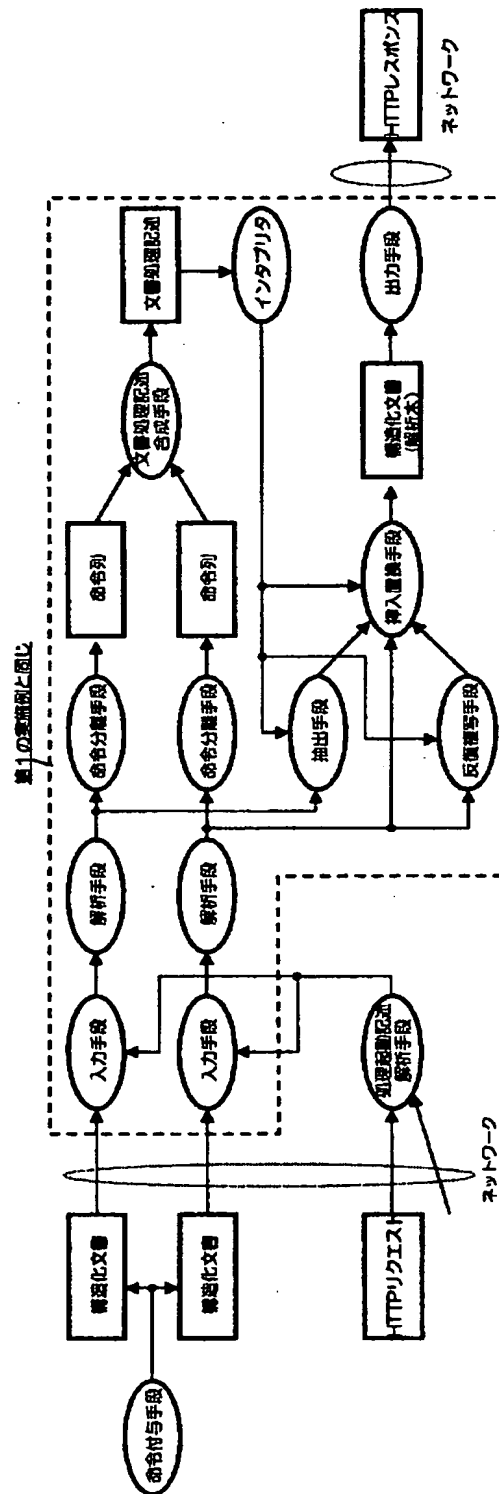




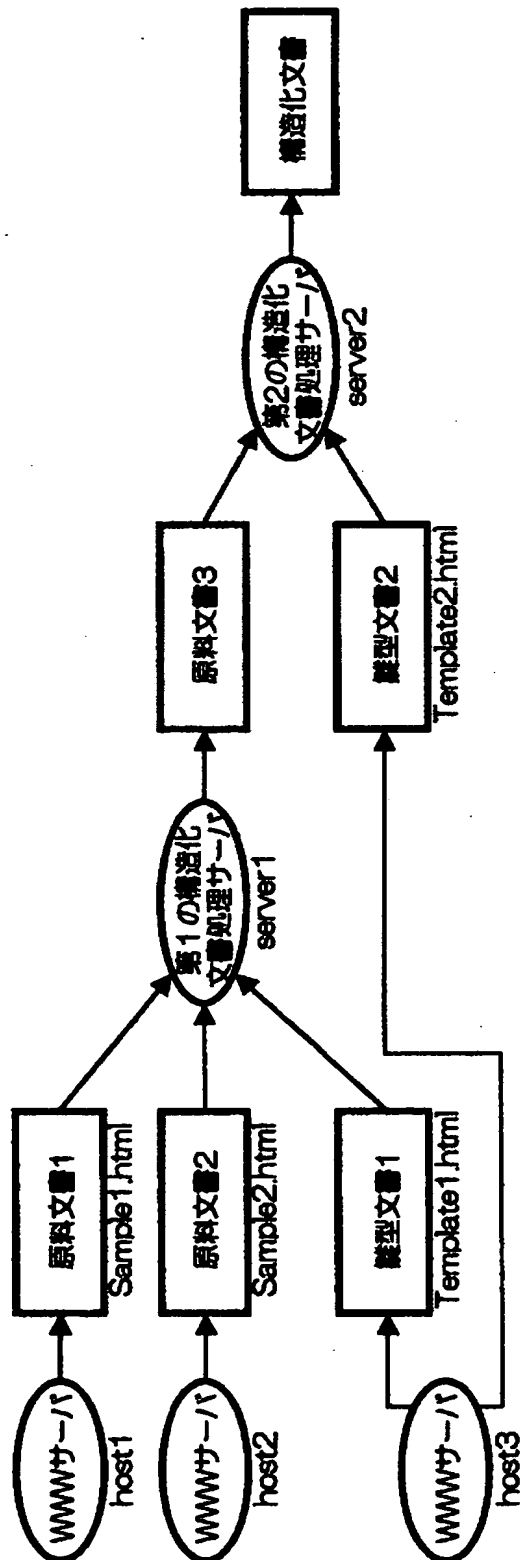
【図 1 2】



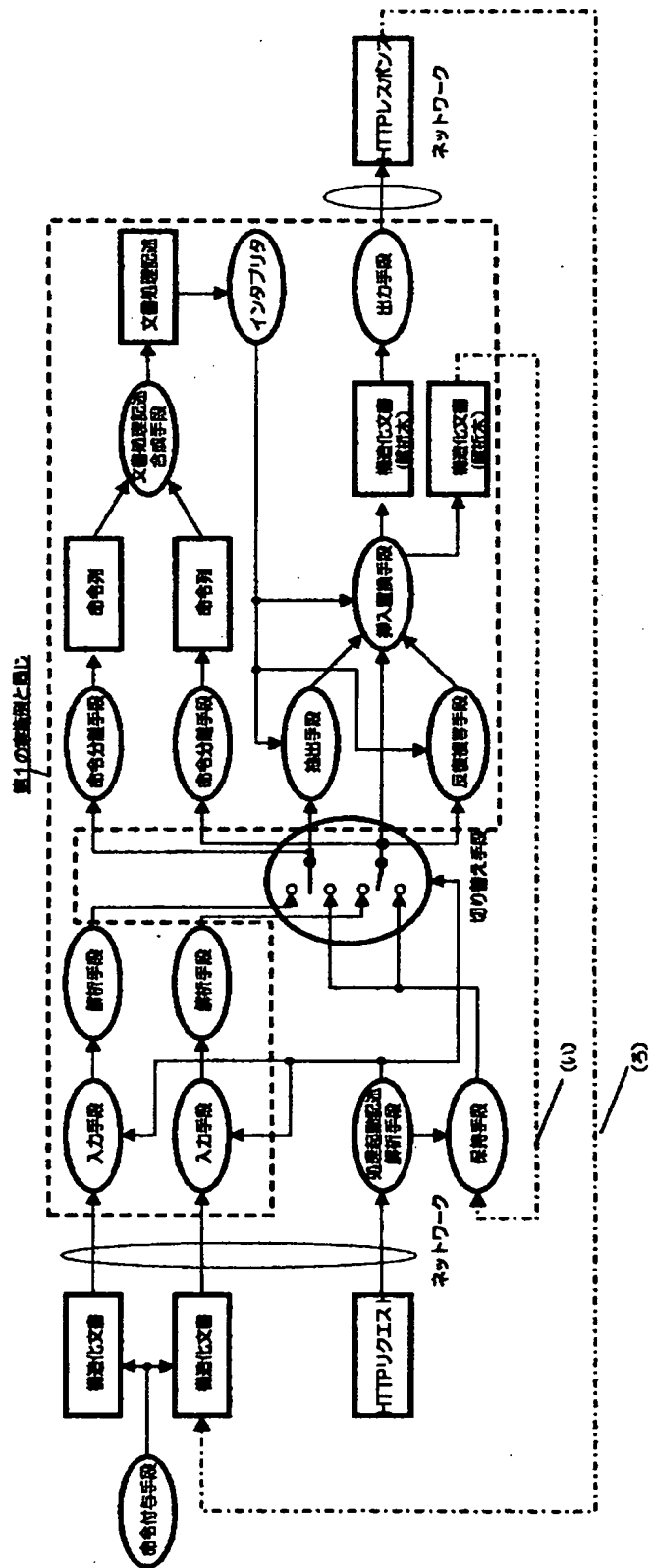
【図 13】



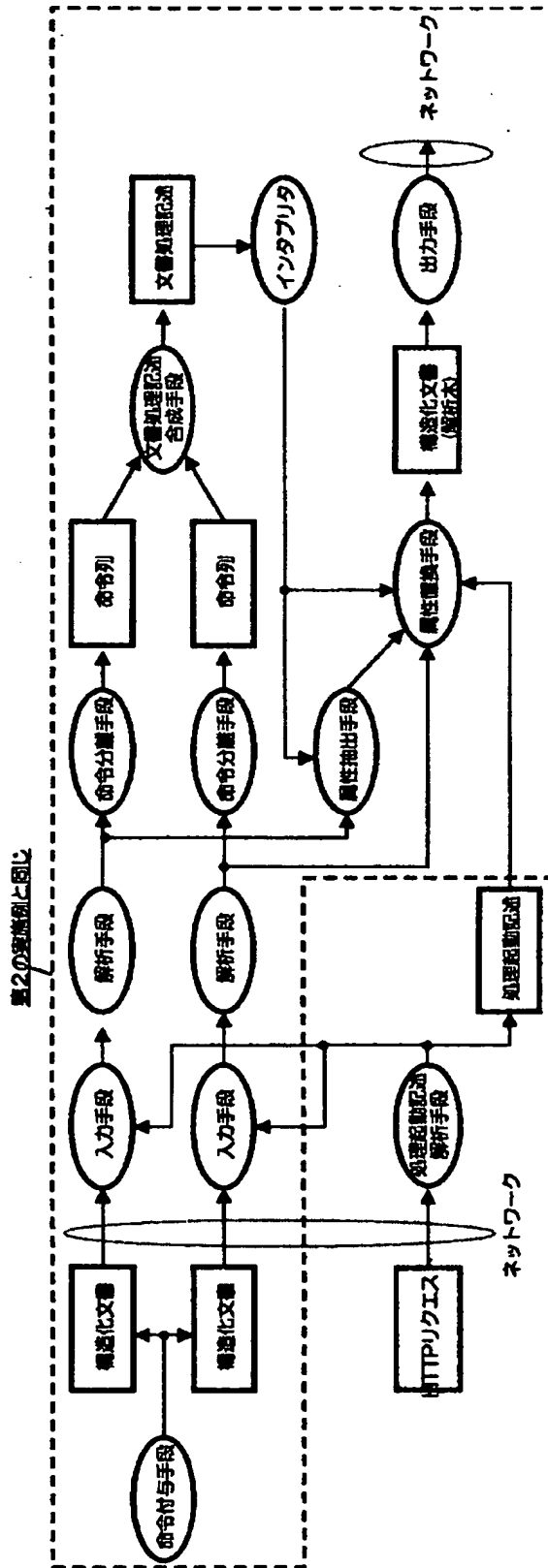
【図 1 4】



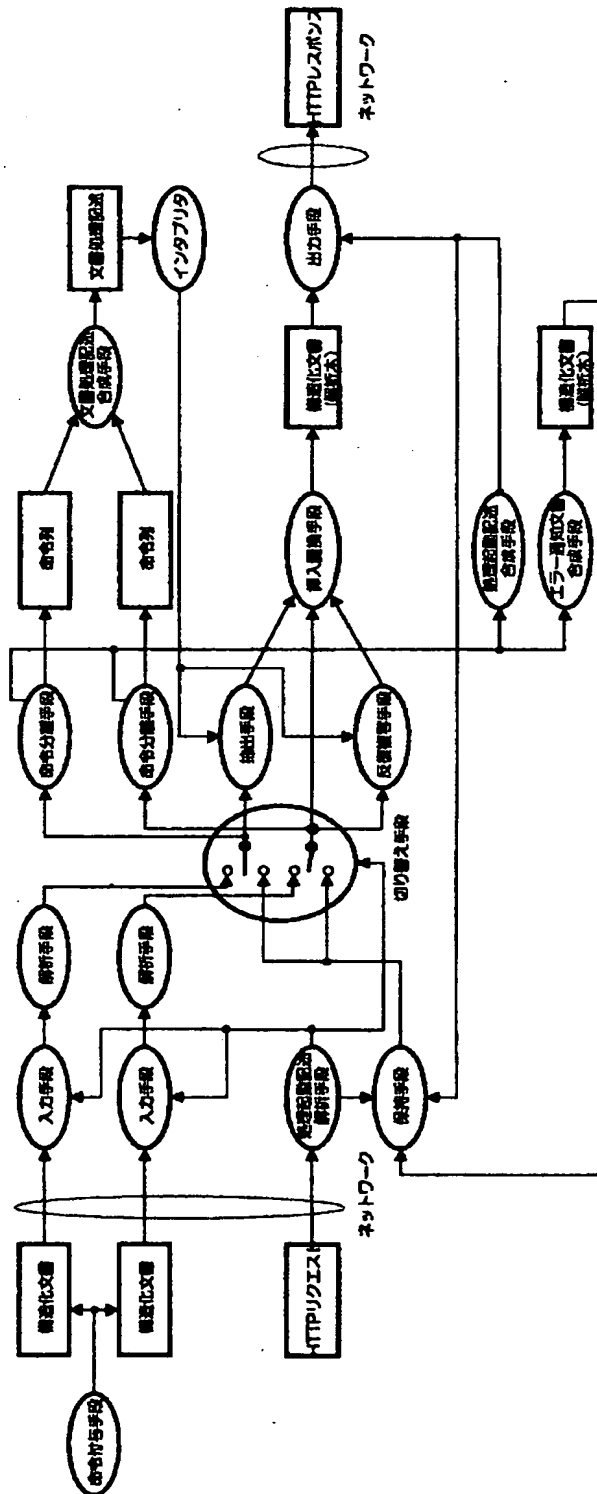
【図 15】



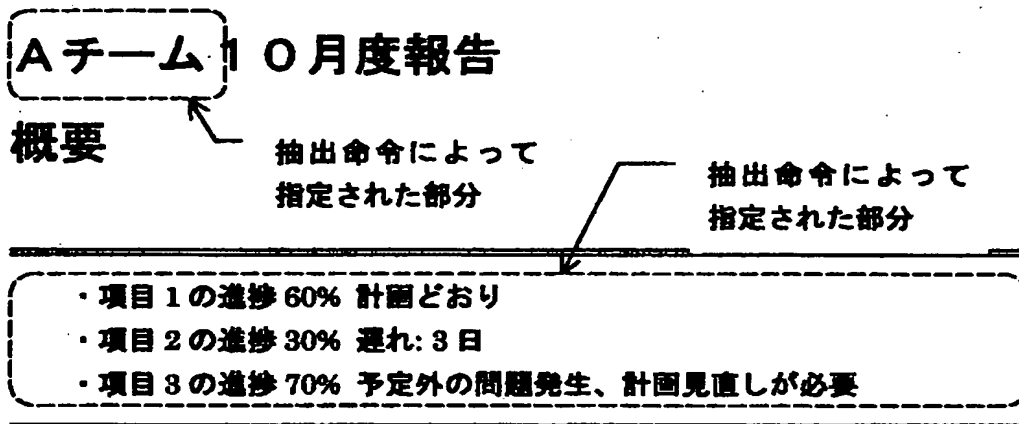
【図 16】



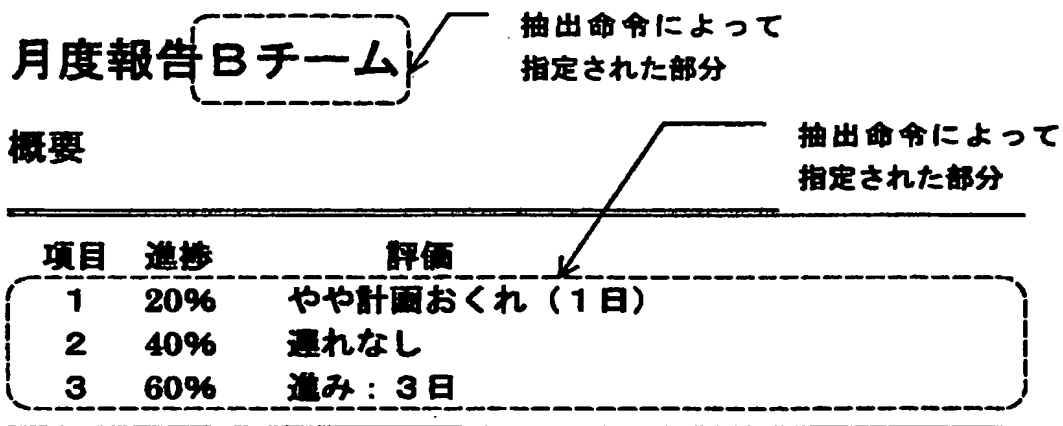
【図 17】



【図18】



【図19】



【図20】

## 進捗サマリ

チーム	概況												
Aチーム	<ul style="list-style-type: none"><li>・項目1の進捗 60% 計画どおり</li><li>・項目2の進捗 30% 遅れ: 3日</li><li>・項目3の進捗 70% 予定外の問題発生、計画見直しが必要</li></ul>												
Bチーム	<table><tr><th>項目</th><th>進捗</th><th>評価</th></tr><tr><td>1</td><td>20%</td><td>やや計画おくれ(1日)</td></tr><tr><td>2</td><td>40%</td><td>遅れなし</td></tr><tr><td>3</td><td>60%</td><td>進み: 3日</td></tr></table>	項目	進捗	評価	1	20%	やや計画おくれ(1日)	2	40%	遅れなし	3	60%	進み: 3日
項目	進捗	評価											
1	20%	やや計画おくれ(1日)											
2	40%	遅れなし											
3	60%	進み: 3日											

【図21】

## 進捗サマリ

チーム	Aチーム	Bチーム												
概況	<ul style="list-style-type: none"> <li>・項目1の進捗 60% 計画どおり</li> <li>・項目2の進捗 30% 遅れ: 3日</li> <li>・項目3の進捗 70% 予定外の問題発生、計画見直しが必要</li> </ul>	<table border="1"> <thead> <tr> <th>項目</th><th>進捗</th><th>評価</th></tr> </thead> <tbody> <tr> <td>1</td><td>20%</td><td>やや計画おくれ(1日)</td></tr> <tr> <td>2</td><td>40%</td><td>遅れなし</td></tr> <tr> <td>3</td><td>60%</td><td>進み: 3日</td></tr> </tbody> </table>	項目	進捗	評価	1	20%	やや計画おくれ(1日)	2	40%	遅れなし	3	60%	進み: 3日
項目	進捗	評価												
1	20%	やや計画おくれ(1日)												
2	40%	遅れなし												
3	60%	進み: 3日												



【図 2 2】

## リンク集

---

---

リンク 1

---

---

【図 2 3】

## 資料集

---

---

チャート
------

---

---

【図24】

## 進捗サマリ

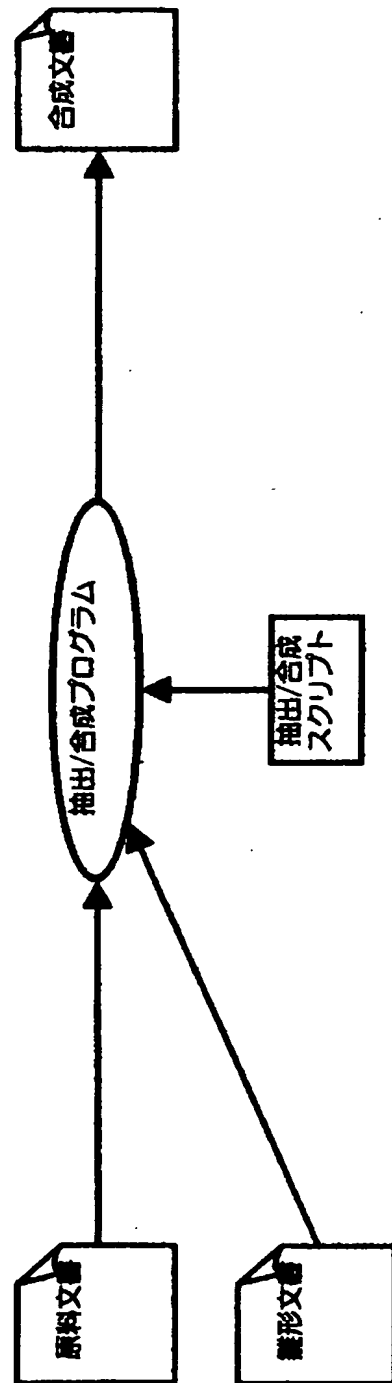
チーム	概況
Aチーム	<ul style="list-style-type: none"> <li>・項目1の進捗 60% 計画どおり</li> <li>・項目2の進捗 30% 遅れ: 3日</li> <li>・項目3の進捗 70% 予定外の問題発生、計画見直しが必要</li> </ul>
Bチーム	<p>項目 進捗 評価</p> <p>1 20% やや計画おくれ(1日)</p> <p>2 40% 遅れなし</p> <p>3 60% 進み: 3日</p>

【図25】

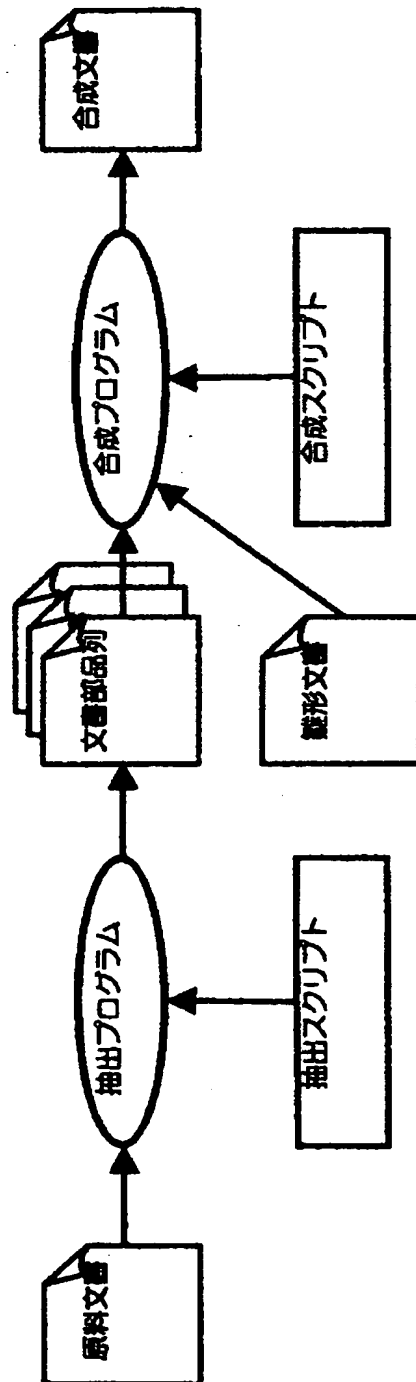
## 進捗サマリ

チーム	概況
Aチーム	<ul style="list-style-type: none"> <li>・項目1の進捗 60% 計画どおり</li> <li>・項目2の進捗 30% 遅れ: 3日</li> <li>・項目3の進捗 70% 予定外の問題発生、計画見直しが必要</li> </ul>
Bチーム	<p>項目 進捗 評価</p> <p>1 20% やや計画おくれ(1日)</p> <p>2 40% 遅れなし</p> <p>3 60% 進み: 3日</p>

【図 2 6】



【図 27】



【書類名】 要約書

【要約】

【課題】 手続きを記述したスクリプトを用いることなしに、構造文書から文書部品を抽出し、各文書部品を雛型文書中に挿入・置換して、文書合成処理を行う。

【解決手段】 構造化文書中には、文書部品を取り出す抽出命令と、反復複写と挿入置換命令が付与されている。したがって、文書部品の取り出しの指定と、反復複写の指定と、文書部品を挿入または置換する文書部品（個所）の指定を行ない、入力した複数の構造化文書から取り出した命令を動的に合成して文書処理記述を作成するようにした結果、文書処理記述スクリプトが不要となる。したがって、元の文書と別にスクリプトを管理するという手間が省略される。

【選択図】 図1



特 2000-018059

出 願 人 履 歴 情 報

識別番号 [000005496]

1. 変更年月日 1996年 5月29日  
[変更理由] 住所変更  
住 所 東京都港区赤坂二丁目17番22号  
氏 名 富士ゼロックス株式会社